Capítulo 2

Introdução à criptografia pós-quântica

Paulo S. L. M. Barreto[†], Felipe Piazza Biasi[†], Ricardo Dahab[*], Julio César López-Hérnandez[*], Eduardo Morais[*], Ana D. Salina de Oliveira[‡], Geovandro C. C. F. Pereira[†], Jefferson E. Ricardini[†]

[†] Universidade de São Paulo
{pbarreto, fbiasi, geovandro, jricardini}@larc.usp.br

[*] Universidade Estadual de Campinas
{rdahab, jlopez, emoraes}@ic.unicamp.br

[‡] Universidade Federal de Matro Grosso do Sul
anakarina@facom.ufms.br

Resumo

Em 1997, Peter Shor publicou um algoritmo quântico capaz de fatorar inteiros grandes e de calcular logaritmos discretos em corpos finitos em tempo hábil. Tais problemas estão na base da segurança de técnicas convencionais de criptografia assimétrica (e.g. RSA, ECC). Isso significa que uma informação criptografada nos dias de hoje não necessariamente estará segura em um momento futuro, caso computadores quânticos tornem-se uma realidade. Felizmente, conjectura-se que alguns esquemas criptográficos baseados em outros problemas computacionais resistem ao ataque de Shor com computadores quânticos e ficaram conhecidos como criptossistemas pós-quânticos, como é o caso de criptossistemas baseados em reticulados, em códigos corretores de erro, sistemas multivariados quadráticos e funções de hash. O objetivo deste minicurso é introduzir noções básicas das principais linhas de pesquisa pós-quântica, bem como apresentar os estudos mais recentes de melhorias dos esquemas, relacionadas a tamanhos de chaves, overhead de assinaturas e criptogramas.

2.1. Introdução

Em meados de 1997, Peter Shor introduziu novas preocupações à criptografia ao descobrir um algoritmo quântico capaz de fatorar inteiros grandes e de calcular logaritmos

discretos em corpos finitos em tempo hábil [Shor 1997]. Isso se deve ao fato de que a segurança de técnicas convencionais de criptografia assimétrica é baseada justamente nesses problemas ou relacionados (e.g. RSA, ECC) [Rivest et al. 1978, Miller 1986]. Desse modo, a segurança efetiva de tais técnicas fica condicionada à construção de um computador quântico de grande porte. Isso significa que uma informação criptografada nos dias de hoje não necessariamente estará segura em um momento futuro por conta da dependência mencionada.

Outra ameaça ainda mais efetiva são as recentes descobertas de algoritmos clássicos, executados em computadores tradicionais, que são capazes de resolver certos logaritmos discretos usados em criptografia assimétrica [Barbulescu et al. 2013].

Felizmente, conjectura-se que alguns esquemas criptográficos baseados em outros problemas computacionais resistem ao ataque de Shor com computadores quânticos e ficaram conhecidos como criptossistemas pós-quânticos; este é o caso de criptossistemas baseados em reticulados [Goldreich et al. 1997], códigos corretores de erro [McEliece 1978, Niederreiter 1986], sistemas multivariados quadráticos ($\mathcal{MQ}$) [Ding e Schmidt 2005, Kipnis et al. 1999] e funções de hash [1] [Ding e Schmidt 2005, Dods et al. 2005a], além das construções de criptografia simétrica em geral.

Por outro lado, o principal desafio em criptografia assimétrica pós-quântica é a redução no tamanho das chaves públicas e privadas, além do *overhead* de espaço considerável por mensagem a ser assinada ou encriptada. Nesse sentido, há um esforço em pesquisa [Bernstein et al. 2008a] para tornar essas técnicas mais eficientes e competitivas com técnicas convencionais, em relação às métricas mencionadas. Vale ressaltar que, em se tratando do tempo de processamento, tamanho de código fonte e ocupação de memória RAM, muitos esquemas pós-quânticos já são competitivos e muitas vezes superam esquemas convencionais.

Tendo em vista o novo paradigma de Internet das coisas, onde objetos quaisquer são dotados de capacidade computacional própria e capazes de conectar-se à internet, podendo atualizar-se de forma autônoma e estabelecer redes interconectadas. Um efeito colateral dessa interconectividade é uma possível vulnerabilidade destes sistemas embarcados. Ataques que têm sido primariamente voltados a PCs podem, ser lançados contra carros, aparelhos celulares, *e-tickets*, RFIDs.

Neste cenário, os dispositivos são caracterizados por apresentarem tipicamente escassez no fornecimento de energia (via bateria) e capacidade limitada de processamento, armazenamento e muitas vezes canais de comunicação com baixa largura de banda (e.g. SMS).

Uma vez que sistemas embarcados são normalmente implantados em larga escala, os custos tornam-se uma das principais preocupações dos projetistas. Logo, soluções de segurança para embarcados devem apresentar baixo custo, que pode ser atingido com o projeto de soluções que minimizem *overhead* de transmissão, processamento e ocupação de memória. Nesse sentido as técnicas de criptografia simétrica em geral já atendem as

[1] O termo “resumo"(criptográfico), em vez de “hash", tem sido usado sem grande aceitação. Por isso, optamos por “funções de hash"ou simplesmente “hash". Dependendo do contexto, hash pode significar também o valor da função de hash num dado argumento. Como plural, usamos “hashes”.

métricas necessárias, sendo a criptografia assimétrica o gargalo na maior parte dos casos.

Primitivas criptograficas assimétricas para encriptação e assinatura digital são essenciais dentro de um arcabouço de segurança moderno. Mas as técnicas convencionais não são suficientemente eficientes em certos aspectos, o que dificulta sua utilização em plataformas embarcadas de baixo poder computacional. Nesse contexto a ausência de operações custosas (operações com inteiros grandes, principalmente exponenciações modulares) das técnicas pós-quânticas as torna mais atraentes em cenários de recursos computacionais escassos, como os descritos anteriormente.

O objetivo do minicurso é introduzir noções básicas das principais linhas de pesquisa pós-quântica (códigos corretores de erros, sistemas $\mathcal{MQ}$, reticulados e *hash*), bem como apresentar os estudos mais recentes visando a melhorias dos esquemas relacionadas a tamanhos de chaves, overhead de assinaturas e criptogramas.

2.2. Esquemas de assinatura digital baseados em funções de hash

Esquemas de assinatura digital baseados em funções de hash popularizaram-se após o trabalho de Ralph Merkle [Merkle 1979] em 1979. O esquema proposto por Merkle *(MSS)* é inspirado no esquema de assinatura *one-time* de Lamport e Diffie [Lamport 1979]. A segurança de tais esquemas é baseada na resistência à colisão e na resistência à inversão da função de hash utilizada. O esquema *(MSS)* é considerado prático e resistente aos computadores clássicos e quânticos, pois não se conhece, na literatura, uma maneira de aplicar o algoritmo de Shor neste esquema. A desvantagem dos esquemas de assinatura digital *one-time* é que uma dada chave privada pode ser utilizada para gerar apenas uma assinatura, embora essa assinatura possa ser verificada um número arbitrário de vezes.

2.2.1. Funções de hash

Funções de hash criptográficas são usadas em aplicações de segurança como esquemas de assinatura digital, identificação de dados, derivação de chaves, entre outros. Formalmente, uma função de hash $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ mapeia cadeias binárias m de tamanho finito e arbitrário para cadeias binárias r de tamanho fixo n. Deste modo, $r = h(m)$.

Dado que o conjunto imagem $\{0,1\}^n$ de h é um subconjunto do $\{0,1\}^*$, é fácil perceber que mais de uma mensagem será mapeada para o mesmo hash. Algumas aplicações necessitam que seja computacionalmente inviável que um atacante encontre duas mensagens aleatórias que gerem o mesmo hash, por exemplo, no contexto de assinaturas digitais; outras apenas requerirão que seja computacionalmente inviável encontrar uma mensagem dado o conhecimento do seu hash.

2.2.2. Propriedades

As propriedades criptográficas básicas que as funções de hash devem possuir são: resistência à pré-imagem, resistência à segunda pré-imagem e resistência a colisão.

1. *Resistência à pré-imagem:* dada uma função $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ e um hash r, é computacionalmente inviável encontrar uma cadeia binária m tal que $h(m) = r$.

2. *Resistência à segunda pré-imagem:* dada uma função $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ e uma

mensagem m, é computacionalmente inviável encontrar m' tal que $m' \neq m$ e $h(m') = h(m)$.

3. *Resistência à colisão:* dada uma função $h : \{0,1\}^* \longrightarrow \{0,1\}^n$, é computacionalmente inviável encontrar $m, m' \in M$ tal que $m' \neq m$ e $h(m') = h(m)$.

Outra propriedade desejável para aplicações práticas, é que a função de hash seja eficiente (velocidade, mémoria, energia etc.) quando for implementada em várias plataformas de hardware e/ou software. É fácil ver que uma função que é resistente a colisão é também resistente à segunda pré-imagem, mas a recíproca não necessariamente é verdade.

2.2.3. Construção de funções de hash

O projeto de funções de hash tem sido baseado em diferentes técnicas tais como: encriptadores de bloco [Matyas et al. 1985, Winternitz 1983, Preneel 1983], o método iterativo de Merkle-Damgård [Merkle 1979] (*MD5*, *SHA-1* e *SHA-2*), a construção esponja [Bertoni et al. 2007] (SHA-3), e primitivas aritméticas [Contini et al. 2005].

Os padrões baseados nessas funções vêm evoluindo, principalmente em função das sucessivos ataques anunciados na literatura e eventos especializados. Recentemente, uma competição pública para escolher o padrão SHA-3 foi concluído, tendo sido vencedora uma função do tipo esponja. Não é nosso objetivo neste texto detalhar tais construções.

2.2.4. Esquemas de Assinatura

Um esquema de assinatura $SIGN$ é uma tripla de algoritmos: $(GEN; SIG; VER)$ que satisfaz as seguintes propriedades:

1. O algoritmo de geração de chaves GEN recebe como entrada um parâmetro de seguranca 1^n e produz um par de chaves (X, Y), onde X é a chave privada e Y é a chave pública.

2. O algoritmo de geração de assinatura SIG recebe como entrada uma mensagem $M \in \{0,1\}^*$ e uma chave privada X e produz uma assinatura Sig, denotada por $Sig \leftarrow SIG_X(M)$.

3. O algoritmo de verificação de assinatura VER recebe como entrada uma mensagem M, uma assinatura Sig de M e uma chave pública Y e produz um bit b, onde $b = 1$ significa que a assinatura é válida e $b = 0$ indica que a assinatura é inválida.

2.2.5. Assinatura one-time

Assinaturas One-Time aparecem inicialmente nos trabalhos Lamport [Lamport 1979] e Rabin [Rabin 1978]. Merkle [Merkle 1979] propôs uma técnica que permite transformar um esquema de assinaturas *one-time* em um esquema com número de assinaturas arbitrário. A seguir descreveremos os esquemas de Lamport [Lamport 1979] e Winternitz [Merkle 1987].

2.2.5.1. Esquema de assinatura one-time de Lamport:

O esquema de assinatura *one-time* de Lamport *(LD-OTS)* foi proposto em [Lamport 1979]. Seja n um inteiro positivo, o parâmetro de segurança, o esquema *LD-OTS* utiliza uma função *one-way*

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$

e uma função de hash criptográfico

$$g : \{0,1\}^* \rightarrow \{0,1\}^n.$$

Geração do par de chaves LD-OTS: A chave de assinatura X consiste de $2n$ cadeias de bits de comprimento n escolhidos aleatoriamente,

$$X = (x_0[0], x_0[1], ..., x_{n-1}[0], x_{n-1}[1]) \quad \in \quad {}_R\{0,1\}^{(n,2n)}.$$

A chave de verificação Y é

$$Y = (y_0[0], y_0[1], ..., y_{n-1}[0], y_{n-1}[1]) \quad \in \quad \{0,1\}^{(n,2n)}.$$

onde $y_i[j] = f(x_i[j])$, $0 \leq i \leq n-1$, $j = 0, 1$.

Geração de assinatura LD-OTS: O hash d de uma mensagem M é assinada usando a chave de assinatura X. Para assinar, primeiro calcula-se $d = g(M) = (d_0, ..., d_{n-1})$. Então, o assinante gera a assinatura

$$Sig = (sig_0, ..., sig_{n-1}) = (x_0[d_0], ..., x_{n-1}[d_{n-1}]) \quad \in \quad \{0,1\}^{(n,n)}.$$

Para gerar Sig, não há aplicações da função de hash, pois a assinatura realiza somente a seleção de bits da chave X de acordo com os bits do hash da mensagem.

Verificação de assinatura LD-OTS: Na verificação de assinatura, o verificador tem como entrada: a assinatura $Sig = (sig_0, ..., sig_{n-1})$, a mensagem M e a correspondente chave pública de verificação Y. Para verificar, primeiro calcula-se o hash da mensagem $d = g(M) = (d_0, ..., d_{n-1})$. Então o verificador checa se

$$Sig = (f(sig_0), ..., f(sig_{n-1})) = (y_0[d_0], ..., y_{n-1}[d_{n-1}]).$$

Na Figura 2.1 ilustramos o esquema de Lamport. Neste exemplo a função *one-way* utilizada foi $f(x) = x + 1 \bmod 16$. Para a geração da chave de verificação, são necessárias $2n$ aplicações da função *one-way*, uma para cada elemento de X. Para a verificação da assinatura, a função *one-way* é executada n vezes, uma para cada elemento de Sig.

2.2.5.2. Esquema de assinatura *one-time* de Winternitz:

Winternitz propôs uma melhora no esquema de assinatura *one-time* de Lamport, diminuindo o tamanho da chave pública e privada. Este esquema *W-OTS* foi mencionado pela primeira vez em [Merkle 1987]. O esquema *W-OTS* utiliza uma função *one-way*

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$

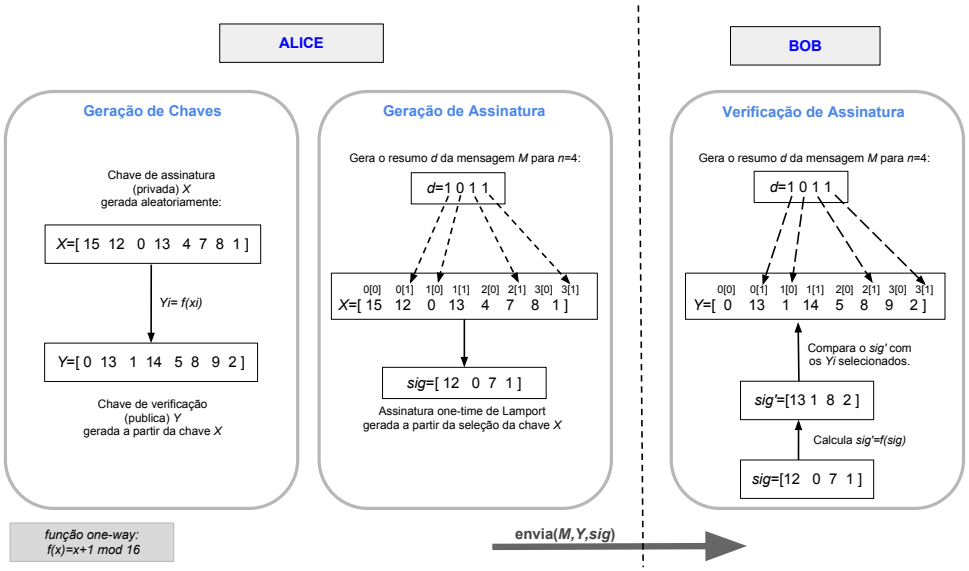


Figura 2.1. Exemplo do esquema de assinatura *one-time* de Lamport

e uma função de hash criptográfica

$$g : \{0,1\}^* \rightarrow \{0,1\}^n,$$

onde n é um inteiro positivo. O parâmetro w denota o número de bits que são processados simultaneamente. Quanto maior for o w menor será a chave de assinatura e maior será o tempo de assinatura e verificação. Em [Dods et al. 2005b] foi feita uma análise comparativa do tempo de execução e tamanho das chaves em relação ao parâmetro w.

Geração do par de chaves W-OTS: Um parâmetro $w \in \mathbb{N}$ é escolhido. A chave de assinatura privada é

$$X = (x_0, ..., x_{t-1}) \in_{\mathcal{R}} \{0,1\}^{(n,t)}$$

onde os x_i são escolhidos aleatoriamente. O tamanho t é obtido calculando $t = t_1 + t_2$, onde

$$t_1 = \left\lceil \frac{n}{w} \right\rceil, \qquad t_2 = \left\lceil \frac{\lfloor log_2 t_1 \rfloor + 1 + w}{w} \right\rceil.$$

A chave pública de verificação

$$Y = (y_0, ..., y_{t-1}) \in \{0,1\}^{(n,t)}$$

é gerada aplicando a função f a cada elemento da chave de assinatura $2^w - 1$ vezes:

$$y_i = f^{2^w-1}(x_i), \qquad para \quad i = 0, ..., t-1.$$

Na Figura 2.2 mostramos um exemplo do processo de geração de chaves para o esquema de assinatura de Winternitz, utilizando uma função *one-way*. Este esquema produz chaves de assinaturas menores que o de Lamport, porém aumenta o número de aplicação da função *one-way* de 1 para $2^w - 1$ em cada elemento da chave de assinatura.

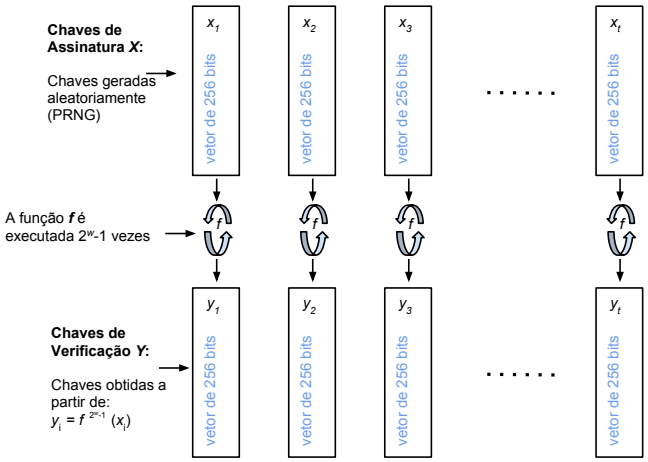


Figura 2.2. Exemplo da geração de chaves no esquema de assinatura *one-time* de Winternitz

Geração de assinatura W-OTS: Para gerar uma assinatura, primeiro calcula-se o hash da mensagem $d = g(M) = (d_0, ..., d_{n-1})$. Se necessário, adicionamos zeros a esquerda de d, tal que d seja divisível por w. Então d é dividido em t_1 blocos binários de tamanho w, resultando em $d = (m_0||...||m_{t_1-1})$, onde $||$ representa a concatenação. Os blocos m_i são representados como inteiros em $\{0, 1, ..., 2^w - 1\}$. Então calcula-se o checksum

$$c = \sum_{i=0}^{t_1-1} (2^w - m_i).$$

Como $c \leq t_1 2^w$, o comprimento da representação binária de c é menor que $\lfloor \log_2 t_1 2^w \rfloor + 1 = \lfloor \log_2 t_1 \rfloor + w + 1$. Se for necessário, adicionam-se zeros à esquerda do c de tal forma que a cadeia c estendida seja divisível por w. Então a cadeia estendida c pode ser dividida em t_2 blocos $c = (c_0||\ldots||c_{t_2-1})$ de comprimento w. Concatenamos em b a cadeia estendida d com a cadeia estendida c. Então $b = (b_0||b_1||\ldots||b_{t-1}) = (m_0||\ldots||m_{t_1-1}||c_0||\ldots||c_{t_2-1})$. Calculamos a assinatura como:

$$Sig = (sig_0, ..., sig_{t-1}) = (f^{b_0}(x_0), f^{b_1}(x_1), ..., f^{b_{t-1}}(x_{t-1})).$$

Verificação de assinatura W-OTS: Para verificar a assinatura $Sig = (sig_0, ..., sig_{t-1})$ da mensagem M, primeiro calculamos os parâmetros $b_0, ..., b_{t-1}$ da mesma forma que durante a geração de assinatura. Depois, calcula-se:

$$sig'_i = f^{2^w-1-b_i}(sig_i), \qquad para \quad i = 0, ..., t-1.$$

Então, para checar a assinatura, verifica:
se $Sig' = (sig'_0, \ldots, sig'_{t-1})$ é igual a $Y = (y_0, \ldots, y_{t-1})$ a assinatura é válida, caso contrário é recusada.

2.2.6. MSS-Esquema de assinatura digital de Merkle

No esquema de assinatura digital de Merkle descrito a seguir, a chave de assinatura e a chave de verificação *one-time* são as folhas da árvore e, a chave pública, é a raiz. Uma árvore com altura H e 2^H folhas terá 2^H pares de chaves *one-time* públicas e privadas.

2.2.6.1. Geração de chaves da árvore de Merkle

Para gerar a chave pública *pub* de Merkle, que corresponde a raiz da árvore de Merkle, primeiro é preciso gerar o par de chaves, pública e privada, *one-time* para cada folha da árvore de Merkle.

Geração do par de chaves one-time: Um algoritmo de assinatura *one-time* gera as chaves privadas $X[u]$ e públicas $Y[u]$, para as folhas $u = 1, ..., 2^H$. O Algoritmo 2.2.1 descreve o processo de geração de um par de chaves *one-time* de Winternitz. Um algoritmo de geração de números pseudo-aleatórios *PRNG* é utilizado para gerar os t elementos da chave de assinatura, reduzindo o custo de armazenamento das chaves privadas, pois somente a semente do *PRNG* é armazenada.

Algoritmo 2.2.1 Geração do par de chaves *one-time* de Winternitz [Merkle 1979]

Entrada: parâmetro Winternitz t.
Saída: Matrizes X, Y de tamanho[2^H][256].

```
Cria as matrizes X e Y.
Cria as matrizes temporárias x e y de tamanho[t][256].
para (u = 1, u < t, u++) faça
    Gera aleatoriamente a semente para o PRNG: X[u] = semente;
    x[0]=PRGN(X[u]);
    para (i = 1, i < t, i++) faça
        y[i-1] = f^(2^w-1)(x[i-1]);
        x[i] = PRNG(x[i-1]);
    Y[u]=g(y[0]||...||y[t-1]);
    retorne X,Y;
```

Geração da chave pública pub: O Algoritmo 2.2.2 gera a chave pública *pub* da árvore de Merkle. Os valores de entrada são: a folha inicial $folhaIni$ e a altura da árvore $alturaMax$. Cada nó folha ($no[u]$) da árvore, recebe a chave de verificação ($Y[u]$) daquela folha. O valor de cada nó intermediário ($no[pai]$) é o resultado da aplicação da função de hash na concatenação dos valores de seus dois filhos, esquerdo ($no[esq]$) e direito ($no[dir]$). Cada vez que uma folha u é calculada e empilhada em *pilhaNo*, o algoritmo verifica se os nós do topo da pilha tem alturas iguais. Se as alturas forem iguais, os dois nós serão desempilhados, concatenados em $no[pai]$ e gerado um novo hash, que será empilhado em *pilhaNo*. O algoritmo termina quando a raiz da árvore é encontrada.

A Figura 2.3 demonstra a ordem em que os nós são empilhados na árvore conforme execução do Algoritmo 2.2.2. Os nós em cinza, representam os nós que já foram gerados. O 4^o nó da árvore gerado (folha $u = 2$), recebeu o valor de $Y[2]$. O 3^o nó é resultado do hash da concatenação dos nós 1 e 2.

Algoritmo 2.2.2 ArvoreDehash [Merkle 1979]

Entrada: Inteiros folhaIni, alturaMax; Matrizes de 256 bits Y;
Saída: Uma estrutura nó pub: A raiz da árvore.

```
Cria uma pilha pilhaNo.
para (u = folhaIni, u < 2^alturaMax, u++) faça
    Guarda Y[u] em nó folha no[u].hash = (Y[u])
    Empilha no[u] na pilha pilhaNo
    enquanto Os nós do topo da pilha pilhaNo tiverem altura iguais faça
        Desempilha o nó no[dir] de pilhaNo
        Desempilha o nó no[esq] de pilhaNo
        Calcula no[pai].hash = g(no[esq].hash||no[dir].hash)
        se no[pai].altura=alturaMax então
            retorne (no[pai])
        senão
            Empilha no[pai] na pilha pilhaNo
```

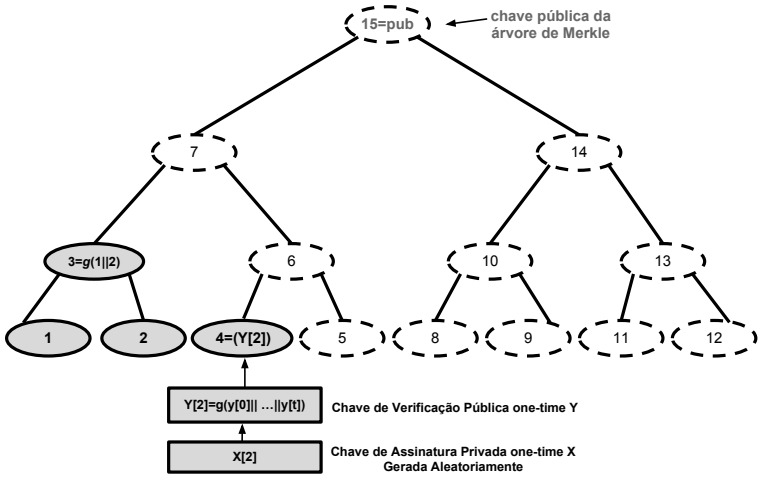


Figura 2.3. Geração da chave pública da árvore de Merkle

2.2.6.2. Geração de assinatura

O esquema *MSS* permite a geração de 2^H assinaturas, para uma árvore de altura H. Suponha que tenhamos $M[u]$ mensagens, para $u = 0,..,2^H$. A mensagem $M[u]$ é assinada com o esquema de assinatura *one-time*, resultando em uma assinatura Sig, no qual utiliza a chave $X[u]$ para assinar a mensagem $M[u]$, conforme apresentado em [J. Buchmann e Szydlo 2008]. Uma árvore de autenticação Aut é utilizada para guardar os nós no caminho necessários para autenticar uma folha $Y[u]$, reduzindo a necessidade de enviar toda a árvore para o receptor. A assinatura SIG de Merkle é composta de Sig e da correspondente chave de verificação $Y[u]$. Também é preciso incluir o índice u (índice da folha) e o respectivo caminho de autenticação, $Aut = (Aut[0],..,Aut[H-1])$. Assim, a assinatura $SIG = (u, Sig, Y[u], (Aut[0],...,Aut[H-1]))$.

O algoritmo clássico do caminho de autenticação. O algoritmo clássico do caminho de autenticação *(Classic Merkle Tree Traversal)* [Merkle 1987] calcula os nós de autenticação Aut para cada folha da árvore, necessários para autenticar a chave pública pub da árvore de Merkle. Este algoritmo utiliza duas variáveis do tipo pilha: Aut e Aux. A pilha Aut contém o caminho de autenticação atual e a pilha Aux guarda os próximos nós de autenticação que serão necessários. O caminho Aut é composto pelos valores dos nós $Aut[h]$ em cada altura que são os irmãos direitos dos nós no caminho de autenticação que ligam a folha até a raiz da árvore de Merkle.

Vamos descrever o processo de geração dos caminhos de autenticação. O primeiro caminho de autenticação é gerado durante a execução do Algoritmo 2.2.2. Os próximos caminhos de autenticação são gerados pelo Algoritmo 2.2.3 a medida em que as assinatura são realizadas. Na Figura 2.4, os nós em cinza mostram o primeiro caminho de autenticação Aut para a folha $u = 0$.

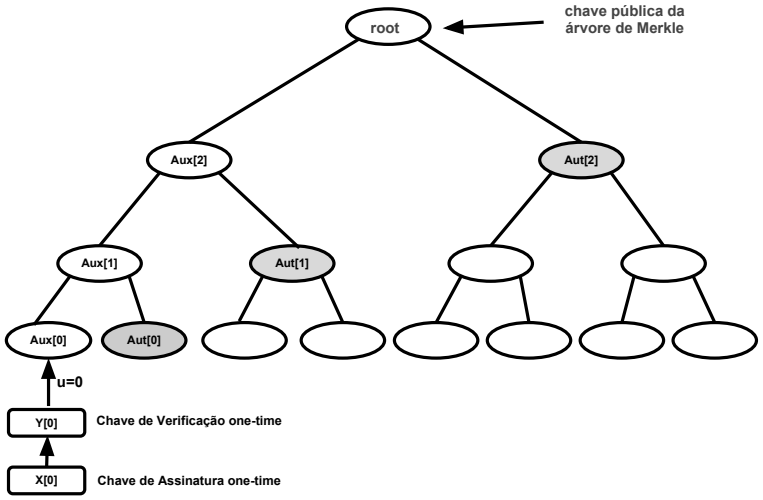


Figura 2.4. Execução do Algoritmo 2.2.2 com o primeiro caminho de autenticação

.

Fase de saída e atualização. O Algoritmo 2.2.3 mostra os passos para produzir o próximo

caminho de autenticação para a folha u seguinte na árvore. A primeira assinatura usa a folha $u = 0$ e a cada assinatura realizada, a folha é atualizada em uma unidade e o próximo caminho de autenticação é preparado de forma eficiente, pois somente os nós do caminho que mudam serão atualizados.

Algoritmo 2.2.3 Algoritmo de percurso na árvore [Merkle 1979]
Entrada: Inteiro altura H.
Saída: Uma assinatura com o caminho de autenticação: Aut.

```
para (u = 0, u < 2^H, u++) faça
    para (h = 0, h < H, h++) faça retorne A pilha Aut da folha u
    O assinante assina com a folha u
    para (h = 0, h < H, h++) faça
        se (u+1)/(2^h) = 0 então
            Atualiza Aut[h] = Aux[h]
            no_Ini = (u + 1 + 2^h) ⊕ 2^h.
            Aux[h].atualiza(no_Ini, h).
```

O Algoritmo 2.2.3 atualiza os nós de autenticação através da execução da função $Aux[h].atualiza(no_{Ini}, h)$, que executa o Algoritmo 2.2.2 no nó e altura selecionados. Depois de 2^h rodadas o valor do nó selecionado será completado.

2.2.6.3. Verificação de assinatura

De acordo com o método em [J. Buchmann e Szydlo 2008], o processo de verificação de assinatura consiste de duas etapas: na primeira etapa, a assinatura Sig é verificada utilizando-se a chave de verificação *one-time* $Y[i]$ e o respectivo algoritmo *one-time*; na segunda etapa, é preciso validar a chave pública da árvore de Merkle, então o receptor calcula o respectivo caminho de autenticação, construindo o caminho $(p[0], ..., p[H])$ da folha i até a raiz, para toda altura h. O índice i é utilizado para decidir em qual ordem o caminho de autenticação será reconstruído. Inicialmente, para a folha de índice i, $p[0] = g(Y[i])$. Para $h = 1, 2, \ldots, H$ executa-se a condição a seguir para calcular o valor hash da sequência dos nós em cada altura:

$$p[h] = \begin{cases} g(Aut[h-1] || p[h-1]) & \text{if } \lfloor i/(2^{h-1}) \rfloor \equiv 1 \bmod 2; \\ g(p[h-1] || Aut[h-1]) & \text{caso contário.} \end{cases}$$

Ao final, compara-se o valor de $p[H]$ com a chave pública conhecida pub. Se o valor for igual, a autenticação é confirmada.

2.2.7. CMSS - Um aperfeiçoamento no esquema de assinatura de Merkle

O esquema *CMSS* [Buchmann et al. 2006] é uma variante do esquema *MSS* que permite aumentar a quantidade de assinaturas de 2^{20} para 2^{40}. Neste trabalho os autores demonstraram que *CMSS* é competitivo na prática, apresentando uma aplicação altamente eficiente dentro do *Java Cryptographic Service Provider FlexiProvider* e mostraram que a aplicação pode ser usada para assinar mensagens no Microsoft Outlook.

No esquema *CMSS* são construídas duas árvores, uma subárvore e uma árvore principal, cada uma com 2^h folhas, para $h = H/2$. Assim, aumenta o número de assinaturas em relação ao *MSS*, pois o *MSS* se torna impraticável para uma altura $H > 25$, porque aumenta muito a quantidade de chaves privadas a serem armazenadas e, a geração de par de chaves leva muito tempo. Para gerar 2^{20} chaves de assinatura, duas árvores com altura 2^{10} são geradas no *CMSS*, enquanto que no *MSS* uma única árvore com 2^{20} nós é construída. Desta forma, o tempo de geração de chaves é reduzido.

Para melhorar o tempo de geração de assinatura, os autores utilizam o algoritmo de Szydlo [Szydlo 2003] que é mais eficiente. Este algoritmo foi implementado no artigo [Buchmann et al. 2008], no qual a proposta é equilibrar a quantidade de cálculos de folhas em cada caminho de autenticação.

Para reduzir o tamanho da chave privada, utiliza-se um gerador de números *PRNG* [NIST 2007] no qual somente a semente do *PRNG* é armazenada. Utilizando uma função de hash de n bits e o parâmetro de Winternitz t, a chave de assinatura teria (tn) bits. Com o uso do gerador *PRNG*, é possível guardar somente a semente de n bits.

O esquema *CMSS* usa duas árvores de autenticação *MSS*, uma árvore principal e uma subárvore, A chave pública *CMSS* é a raiz da árvore principal. Os dados são assinados com as folhas da subárvore. Após as 2^h primeiras assinaturas serem geradas, uma nova subárvore é construída e utilizada para gerar as próximas 2^h assinaturas.

Geração de Chaves CMSS: Para gerar o par de chaves, o algoritmo de geração de chaves do *MSS* é chamado duas vezes. Inicialmente, a primeira subárvore e seu primeiro caminho de autenticação são gerados. Então, a árvore principal e seu primeiro caminho de autenticação são computados. A chave pública *CMSS* é a raiz da árvore principal. *CMSS* usa o esquema de assinatura *one-time* de Winternitz. Observamos o esquema *CMSS* na figura 2.5.

Geração de Assinatura CMSS: A geração de uma assinatura *CMSS* é realizada em quatro partes. Primeiro, a assinatura *MSS* do documento d é calculada usando a subárvore. Então, a assinatura *MSS* da raiz da subárvore é calculada usando uma folha árvore principal. Então, a próxima subárvore é parcialmente construída. Finalmente, a chave privada *CMSS* é atualizada para a próxima assinatura. Cada vez que uma nova assinatura *CMSS* é computada, a assinatura da raiz da subárvore atual é recalculada. O tempo necessário para recalcular esta assinatura *MSS* é tolerável.

Verificação de Assinatura CMSS: A verificação de assinatura *CMSS* é realizada em dois passos. Primeiro, os dois caminhos de autenticação são validados, então a validade das duas assinaturas *one-time* são verificadas.

2.2.8. GMSS - Esquema de Merkle com capacidade virtualmente ilimitada de geração de assinaturas

O esquema *GMSS* foi publicado em 2007 [Buchmann et al. 2007], no qual os autores propõem uma alteração no esquema de assinatura Merkle que permite assinar um número ilimitado de mensagens 2^{80} usando um único par de chaves. O esquema *GMSS* foi

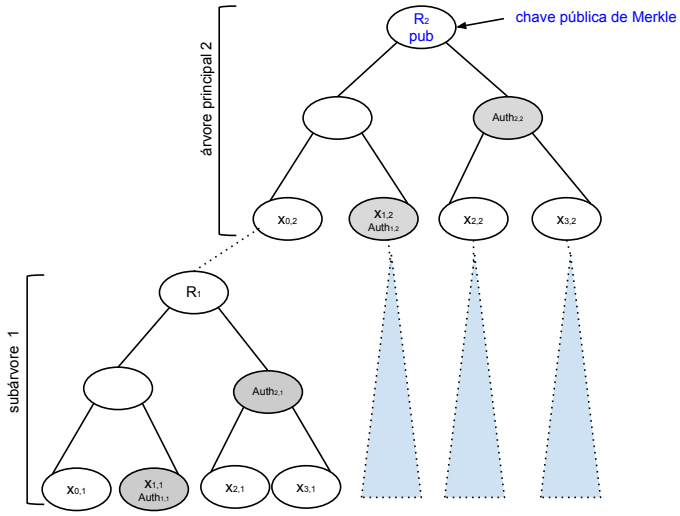


Figura 2.5. Esquema de assinatura CMSS

utilizado para projetar um protocolo cliente servidor para servidores web usando *SSL/TLS* que minimiza a latência e aumenta a resistência à ataques de negação de serviço.

A construção básica de *GMSS* consiste de uma árvore com T camadas. Subárvores em camadas diferentes podem ter alturas diferentes. Para reduzir o custo de geração de uma assinatura, o esquema *GMSS* usa a geração de uma assinatura de forma distribuída, distribuindo a geração de assinatura em cada subárvore em várias assinaturas. Este esquema também possibilita escolher parâmetros w de Winternitz diferentes para subárvores em camadas diferentes, produzindo assim, assinaturas menores.

Geração de chaves GMSS: Para cada subárvore, o algoritmo de geração de chaves, *W-OTS*, gera as chaves de assinatura e o Algoritmo 2.2.2 da Árvore de hash calcula as raízes das árvore $Raiz_{\tau,0}$. Os primeiros caminhos de autenticação de cada subárvore são salvos durante a geração da *Raiz*. Depois as assinaturas Sig_τ das T árvores Merkle que serão usadas para a primeira assinatura são calculadas. Como os valores mudam com menos frequência para as camadas superiores, a precomputação pode ser distribuída por várias etapas, resultando em uma melhora significativa da velocidade de assinatura e permitindo escolher grandes parâmetros w de Winternitz, que resulta em assinaturas menores. Para garantir tamanhos pequenos de chaves privadas, foi utilizado *PRNG*, onde somente a semente do *PRNG* precisa ser armazenada.

Geração de assinatura GMSS: A raiz de uma árvore filha é assinada com a chave de assinatura *one-time* correspondente a uma certa folha de sua árvore mãe. $Raiz_\tau$ denota a raiz da árvore τ. Sig_τ denota a assinatura *one-time* de $Raiz_\tau$, que é gerado usando a folha l do pai de τ. Os hashes da mensagem d são assinados utilizando as folhas das árvores Merkle sobre a camada T mais profunda. O número de mensagens que podem ser assinadas com um par de chaves *GMSS* é $S = 2^{h_1+...+h_T}$, onde $h_1, ...h_T$ são as alturas das subárvores. A assinatura *GMSS* consiste do índice das folhas s, das assinaturas *one-time*

Sig_d e $Sig_{\tau_{i,j_i}}$ onde $i = 2,...,T$ e $j = 0,...,2^{h_1+...+h_{i-1}} - 1$ e dos caminhos de autenticação $Aut[\tau_{i,j_i}, l_i]$ para $i = 1,...,T$.

Na geração de assinatura também são calculados as raízes $Raiz_{\tau_{i,1}}$ e caminhos de autenticação $Aut[\tau_{i,1}, 0]$, das árvores seguintes $\tau_{i,1}$, onde $i = 2,...,T$. A geração de assinatura pode ser dividida em duas partes. A primeira parte, parte *online*, calcula Sig_d e a assinatura. A segunda parte, parte *offline*, precalcula os caminhos de autenticação e assinaturas *one-time* das raízes necessárias para as próximas assinaturas.

Verificação de assinatura GMSS: O processo de verificação de *GMSS* é essencialmente o mesmo que para *MSS* e *CMSS*. Primeiro, verifica-se a assinatura de uma só vez Sig_d do hash d usando o esquema *one-time* de Winternitz. Então, o verificador valida as chaves públicas das raízes das subárvores e da árvore principal.

2.2.9. XMSS - *eXtended Merkle Signature Scheme*

O esquema de assinatura *XMSS* [Buchmann et al. 2011b] é uma modificação do esquema *MSS*, sendo o primeiro esquema de assinatura prático comprovadamente *foward secure* com exigências mínimas de segurança. Este esquema utiliza uma família de funções F e uma família de funções de hash G. O esquema *XMSS* é eficiente, se G e F são eficientes. Este esquema, é infalsificável sob ataques adaptativos de mensagem escolhida no modelo padrão se G é resistente à segunda pré-imagem e F é pseudo aleatório. Os parâmetros de *XMSS* são: o parâmetro de segurança $n \in \mathbb{N}$, o parâmetro de Winternitz $w \in \mathbb{N}(w > 1)$, o tamanho da mensagem em bits $m \in \mathbb{N}$, uma família de funções pseudo-aleatórias:

$$F_n = \{f_K : \{0,1\}^n \rightarrow \{0,1\}^n | K \in \{0,1\}^n\},$$

a altura da árvore $H \in \mathbb{N}$, uma função de hash g_K escolhida aleatoriamente com distribuição uniforme da família de funções:

$$G_n = \{g_K : \{0,1\}^{2n} \rightarrow \{0,1\}^n | K \in \{0,1\}^n\}$$

e a chave de assinatura *one-time* $x \in \{0,1\}^n$ escolhida aleatoriamente com distribuição uniforme.

A chave de assinatura *one-time* x é utilizada para construir a chave de verificação *one-time* y, através da aplicação da família de funções F_n. Neste trabalho utilizou-se a função $f_K(x) = g(Pad(K)||Pad(x))$, para uma chave $K \in \{0,1\}^n$, $x \in \{0,1\}^n$ e $Pad(z) = (z||10^{b-|z|-1})$ para $|z| < b$, sendo b o tamanho do bloco da função de hash.

O esquema *XMSS* utiliza uma versão um pouco modificada do esquema de Winternitz [Buchmann et al. 2011a]. Esta modificação permite eliminar a necessidade de uma família de funções de hash resistente à colisão. Utiliza um caminho através da família de funções em vez de uma avaliação iterada da função de hash. Para $K, x \in \{0,1\}^n$, $e \in \mathbb{N}$, e $f_K \in F_n$, define-se a função $f_K^e(x)$ como: $f_K^0(x) = K$, e para $e > 0$ $f_K^e(x) = f_{K'}(x)$, onde $K' = f_K^{e-1}(x)$.
Para o parâmetro w de Winternitz, calcula-se:

$$l_1 = \left\lceil \frac{m}{log_2(w)} \right\rceil, \qquad l_2 = \left\lfloor \frac{log_2(l_1(w-1))}{log_2(w)} \right\rfloor + 1, \qquad l = l_1 + l_2.$$

A chave pública de verificação é:

$$Y = (y_1, \ldots, y_l) = (f_{sk_1}^{w-1}(x), \ldots, f_{sk_l}^{w-1}(x)).$$

O esquema *W-OTS* assina mensagens binárias de comprimento m. Os bits da mensagem são processados na base w. A mensagem é da forma M = (m_1,...,m_{l_1}), $m_i \in \{0, ..., w-1\}$. A soma de verificação é $c = \sum_{i=1}^{l_1}(w - 1 - m_i)$ em base de representação w é anexada a M. A soma de verificação é de comprimento l_2. O resultado da concatenação dos blocos de M com c é $b = (b_1, ..., b_l)$. O tamanho da assinatura, das chaves de assinatura e verificação é de l cadeias de n bits . A assinatura é:

$$Sig = (sig_1, ..., sig_l) = (f_{sk_1}^{b_1}(x), ..., f_{sk_l}^{b_l}(x)).$$

A árvore *XMSS* é uma modificação da árvore de hash de Merkle. A árvore de altura H, tem $H + 1$ níveis. Utiliza função de hash g_K e vetores de *bitmasks* $(b_{l,j}||b_{r,j}) \in \{0,1\}^{2n}$, escolhidos aleatoriamente, sendo $b_{l,j}$ o *bitmask* esquerdo e $b_{r,j}$ o *bitmask* direito. Os nós no nível j, $0 \leq j \leq H$, são denotados por $NO_{i,j}$, $0 \leq i < 2^{H-j}$, e $0 < j \leq H$. Os nós são calculados como:

$$NO_{i,j} = g_K((NO_{2i,j-1} \oplus b_{l,j})||(NO_{2i+1,j-1} \oplus b_{r,j})).$$

Os bitmasks são a principal diferença das outras árvores de Merkle, pois eles permitem substituir a resistência à colisão da família função de hash. Podemos observar na Figura 2.6 como os nós $NO_{i,j}$ da árvore do esquema *XMSS* são construídos em cada nível j para gerar a chave pública da árvore.

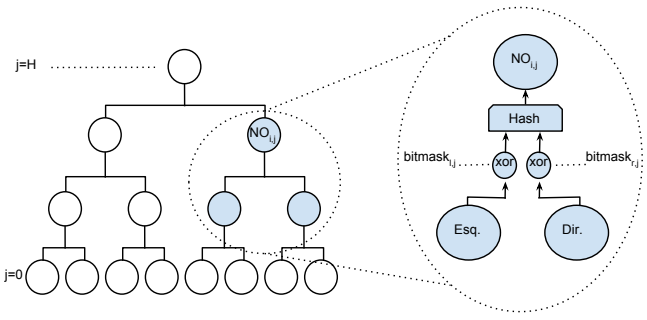


Figura 2.6. Esquema de assinatura XMSS

.

2.2.10. Segurança dos esquemas de assinatura baseados em funções de hash

Nesta seção apresentamos os principais resultados conhecidos sobre a segurança dos esquemas de assinatura baseados em funções de hash.

Definimos agora o conceito de *existencialmente infalsificável sob ataque adaptativo de mensagens escolhidas (existential unforgeability under adaptive chosen message*

attack) de um esquema $SIGN$, onde $SIGN = (GEN, SIG, VER)$ é um esquema de assinatura e (X, Y) um par de chaves gerado por GEN. Este modelo de segurança assume a existência de um poderoso falsificador. O falsificador tem acesso à chave pública e a um oráculo $O(X, .)$ que, por sua vez, tem acesso à chave privada. Passa-se ao oráculo uma mensagem, e o oráculo retorna a assinatura desta mensagem. O falsificador escolhe, no máximo, q mensagens e permite que o oráculo encontre as assinaturas dessas mensagens. As consultas ao oráculo são adaptativas pois uma mensagem pode depender das respostas do oráculo às mensagens anteriormente consultadas. A saída do falsificador é um par (M', Sig'). O falsificador ganha se M é diferente de todas as mensagens consultadas junto ao oráculo e se $VER(M', Sig', Y) = 1$. O esquema de assinatura SIGN é (t, ϵ, q) existencialmente infalsificável sob ataque adaptativo de mensagem escolhida se para qualquer falsificador que roda em tempo t, a probabilidade de sucesso para vencer o jogo acima é no máximo ϵ. Se $SIGN$ tem esta característica, é também chamado de esquema de assinatura (t, ϵ, q). Para assinaturas *one-time*, $q = 1$, pois a chave de assinatura *one-time* pode ser usada apenas uma vez. Para o esquema de assinatura de Merkle, nós temos que $q \leqslant 2^H$.

No trabalho [J. Buchmann e Szydlo 2008] foi provado que o esquema de assinatura one-time Lamport-Diffie é existencialmente infalsificável sob um ataque adaptativo de mensagem escolhida ($CMAseguro$) assumindo a função *one-way* é resistente à preimagem. No mesmo trablho [J. Buchmann e Szydlo 2008] foi provado que o esquema de assinatura Merkle é existencialmente infalsificável quando a função de hash é resistente à colisão e o esquema de assinatura *one-time* subjacente é existencialmente infalsificável.

Uma família de funções de hash resistente à colisão (t_{CR}, ϵ_{CR}) e resistente à préimagem (t_{OW}, ϵ_{OW}). O nível de segurança do MSS é calculado conforme abaixo:

$$\epsilon \rightarrow 2 * max\{\epsilon_{CR}, 2^H * 4n * \epsilon_{OW}\}$$

$$t = min\{t_{CR}, t_{OW}\} - 2^H * t_{SIG} - t_{VER} - t_{GEN}$$

$$t_{GEN} \rightarrow 2^H * 6n, t_{SIG} \rightarrow 4 * n(H + 1), t_{VER} \rightarrow n + H$$

Nas análises efetuadas em [J. Buchmann e Szydlo 2008] para computadores clássicos, foram consideradas que as funções de hash com saída de tamanho n somente admitem ataques genéricos contra suas pré imagens e resistência à colisão. Esses ataques são o de busca exaustiva e o ataque de aniversário. Quando computadores clássicos são usados, um ataque de aniversário testa $2^{n/2}$ valores hash com probabilidade de sucesso de aproximadamente $1/2$. Também, uma busca exaustiva de $2^n/2$ cadeias aleatórias gera uma pré-imagem de um dado valor hash com probabilidade $1/2^{n/2}$. Assim, nós assumimos que a família de funções de hash $\mathcal{G}$ é $(2^{n/2}, 1/2)$ resistente à colisão e $(2^{n/2}, 1/2^{n/2})$ resistente à préimagem. O nível de segurança do esquema de assinatura de Merkle combinado com o esquema de assinatura one-time Lamport-Diffie é pelo menos

$$b = n/2 - 1$$

se a altura da árvore Merkle é pelo menos $H \rightarrow n/3$ e o tamanho da saída da função de hash é pelo menos $n \geqslant 87$, para computadores clássicos.

2.2.11. Resultados da Implementação em Software

No trabalho [Oliveira e López 2013] foram apresentado os resultados de uma implementação em software dos esquemas esquemas *MSS*, *CMSS*, *GMSS* e *XMSS*, utilizando uma máquina Intel Core $i7-2670$ QMCPU, 2.20 GHz com 6 GB de RAM.

A Tabela 2.1 mostra os tamanhos das chaves em bytes e os tempos de execução dos algoritmos implementados com a função de hash *SHA-2*. Definiu-se: (C_{pub}) tamanho da chave pública, (C_{priv}) tamanho da chave privada, (C_{sig}) tamanho da chave de assinatura, (t_{sigOnl}) tempo de assinatura *Online*, ($t_{sigOffl}$) tempo de assinatura *Offline*. A assinatura *online*, calcula a assinatura *W-OTS* e a assinatura *offline*, precalcula os caminhos de autenticações para a próxima assinatura. Para o esquema *GMSS* com 2 subárvores, definimos *GMSS* T=2 (w_2, w_1) e para 4 subárvores definimos *GMSS T=4* (w_4, w_3, w_2, w_1), sendo o parâmetro w_1 utilizado para a árvore da camada mais acima e w_2,w_3 e w_4 para as subárvores das camadas mais abaixo.

Tabela 2.1. Tamanhos em bytes, tempos em (ms,s,min) e função SHA-2(256)

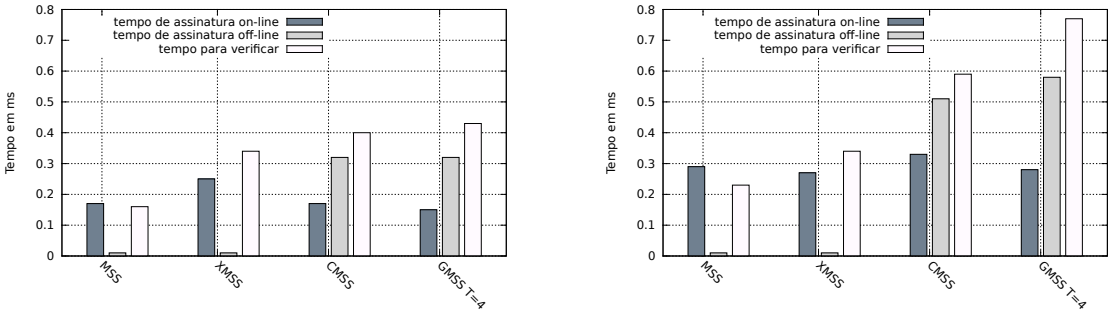
Esquema	H	w	C_{pub}	C_{priv}	C_{sig}	t_{chaves}	t_{sigOnl}	$t_{sigOffl}$	t_{ver}
MSS	20	3	32	1316	3556	243.5 s	0.13 ms	0.01 ms	0.11 ms
MSS	20	4	32	1316	2820	311.5 s	0.17 ms	0.01 ms	0.16 ms
CMSS	20	(3,3)	32	2056	6472	0.30 s	0.13 ms	0.25 ms	0.28 ms
CMSS	20	(4,4)	32	2056	5000	0.37 s	0.17 ms	0.32 ms	0.40 ms
CMSS	40	(3,3)	32	3976	7112	269 s	0.14 ms	0.26 ms	0.30 ms
CMSS	40	(4,4)	32	3976	5640	384 s	0.17 ms	0.32 ms	0.40 ms
GMSS T=2	40	(9,3)	32	3976	5224	48.1 min	0.16 ms	0.27 ms	4.98 ms
GMSS T=4	80	(3,3,3,3)	32	9232	14224	570 s	0.14 ms	0.29 ms	0.45 ms
GMSS T=4	80	(7,7,7,3)	32	9232	10864	50.3 min	0.13 ms	0.25 ms	2.3 ms
XMSS	20	4	1696	1316	4932	293.6 s	0.25 ms	0.01 ms	0.34 ms

Observamos, pela Tabela 2.1, que o tamanho das chaves públicas tem 32 bytes, exceto para o esquema *XMSS* que guarda *bitmasks*. A chave privada e a assinatura são menores nos esquemas *MSS* e *XMSS*, pois nos outros esquemas é preciso guardar informações de 2 ou mais árvores.

Nos gráficos da Figura 2.7 observamos os resultado dos tempos de assinatura e verificação obtidos com $w = 4$, $H = 20$ utilizando as funções *SHA-2* e *SHA-3*. O esquema *MSS* teve os melhores tempos de assinatura e verificação, porque somente um caminho de autenticação precisa ser atualizado e checado para cada assinatura. Porém, o *MSS* é recomendado somente para aplicações que necessitam até 2^{20} chaves de assinatura, tornando-se ineficiente para gerar mais chaves por consumir muitos recursos de memória. Se forem necessárias mais chaves de assinatura, os algoritmos *CMSS* e *GMSS* são recomendados.

2.3. Criptografia de chave pública baseada em sistemas multivariados

Os criptossistemas multivariados de chave pública (*multivariate public key cryptosystems*, ou MPKC) constituem mais uma das principais famílias de criptossistemas de chave pública considerada potencialmente resistentes até mesmo aos poderosos computadores quânticos do futuro. Esquemas MPKC têm sua segurança baseada, em última análise, na dificuldade de resolver sistemas de equações não-lineares multivariadas sobre



(a) SHA-2 (b) SHA-3

Figura 2.7. Tempos de assinatura e verificação com w=4 e H=20

corpos finitos. Em particular, na maior parte dos casos, tais esquemas se baseiam em resolver sistemas de equações multivariadas **quadráticas**. Esse último problema ficou conhecido como o Problema $\mathcal{MQ}$ (multivariate quadratic problem), e foi mostrado ser NP-difícil por Patarin [Patarin e Goubin 1997]. MPKC tem se desenvolvido mais intensamente nas duas últimas décadas. Descobriram que algumas das construções existentes não eram tão seguras quanto se acreditava inicialmente, enquanto outras ainda permanecem viáveis.

A principal ideia em sistemas MPKC é definir uma função de mão única com alçapão que tenha como imagem um sistema não-linear de equações multivariadas sobre um corpo finito. A chave pública é dada por um conjunto de polinômios:

$$\mathcal{P} = \{p_1(x_1, \ldots, x_n), \cdots, p_m(x_1, \ldots, x_n)\}$$

onde cada p_i é um polinômio não-linear (comumente quadrático) nas variáveis $\mathbf{x} = (x_1, \cdots, x_n)$:

$$p_k(x_1, \ldots, x_n) := \sum_{1 \le i \le j \le n} P_{ij}^{(k)} x_i x_j + \sum_{1 \le i \le n} L_i^{(k)} x_i + c^{(k)}, 0 \le k \le m \tag{1}$$

e todos os coeficientes e variáveis estão em $\mathbb{F}_q$. Para simplificar a definição anterior, usaremos notação vetorial, que é inclusive mais próxima de uma implementação:

$$p_k(\mathbf{x}) := \mathbf{x} P^{(k)} \mathbf{x}^{\mathrm{T}} + L^{(k)} \mathbf{x}^{\mathrm{T}} + c^{(k)} \tag{2}$$

onde $P^{(k)} \in \mathbb{F}_q^{n \times n}$ é uma matriz de tamanho $n \times n$, formada com os coeficientes da parte quadrática de $p_k(x_1, \ldots, x_n)$, $L^{(k)} \in \mathbb{F}_q^n$ é um vetor formado com os coeficientes da parte linear de $p_k(x_1, \ldots, x_n)$, e $c^{(k)}$ denota um termo constante de $p_k(x_1, \ldots, x_n)$. $\mathbf{x}$ é o vetor linha das variáveis $[x_1, \ldots, x_n]$. A Figura 2.8 ilustra a transformação (ou mapa) puramente quadrático $\mathbf{x} P^{(k)} \mathbf{x}^{\mathrm{T}}$ (que fornece um certo elemento no corpo finito, denotado por $h_k \in \mathbb{F}_q$) em termos de matrizes e vetores.

A seguir é dada uma definição mais formal para o Problema $\mathcal{MQ}$.

Definição 1 (Problema $\mathcal{MQ}$). *Resolva o sistema $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x}) = \cdots = p_m(\boldsymbol{x}) = 0$, onde cada p_i é quadrático em $\boldsymbol{x} = (x_1, \ldots, x_n)$. Todos os coeficientes e variáveis estão em $K = \mathbb{F}_q$, o corpo de q elementos.*

$$x\,P^{(k)}\,x^T = h_k \;\; (k = 1, \ldots, m)$$

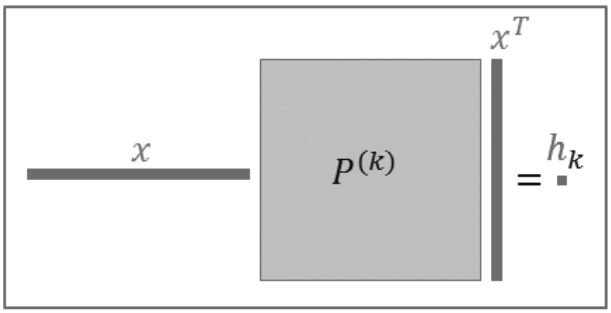


Figura 2.8. Mapa ou Transformação Puramente Quadrática

Em outras palavras, o objetivo do problema $\mathcal{MQ}$ é encontrar uma solução x para um dado mapa $\mathcal{P}$. Garey e Johnson provaram em 1979 [Garey e Johnson 1979, página 251] que a variante decisional do problema $\mathcal{MQ}$ é um problema NP-completo sobre corpos finitos. Mas infelizmente, aplicando-se equivalência polinomial de problemas de busca e decisão para linguagens NP-completas [Bellare e Goldwasser 1994], o resultado de Garey e Johnson fornece apenas uma dificuldade de pior caso do problema $\mathcal{MQ}$ decisional. E, de fato, a complexidade do problema $\mathcal{MQ}$ depende fortemente da relação entre n e m, além da estrutura dos polinômios.

Curiosamente, foi mostrado que o problema $\mathcal{MQ}$ se torna solúvel em *tempo polinomial* usando a técnica de *linearização* para sistemas subdeterminados em corpos de característica par quando $n \geq m(m+1)$ e, em corpos de característica ímpar quando $n \geq 2^{m/7}(m+1)$ [Kipnis et al. 2003, Sec. 7] [Courtois et al. 2002], ou para sistemas sobredeterminados com $m \geq n(n+1)/2$, ou ainda melhor, $m \geq n(n-1)/2$ para o caso especial $\mathbb{F}_2$.

Por outro lado, todos os esquemas de assinatura $\mathcal{MQ}$ propostos até o momento não têm sua segurança baseada somente no problema $\mathcal{MQ}$. Para reconstruir a chave privada, ou ainda encontrar uma equivalente, é necessário resolver um problema relacionado, o Problema do Isomorfismo de Polinômios ou IP, proposto por Patarin [Patarin 1996]. Note que para todos os polinômios de algum grau fixado, todos os argumentos seguintes são polinômios redutíveis a polinômios de grau 2.

Definição 2 (Problema do Isomorfismo de Polinômios (IP)). *Sejam $m, n \in \mathbb{N}$ fixados e arbitrários. Sejam também $\mathcal{P}, \mathcal{Q} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ dois mapas quadráticos e $\mathcal{T} \in \mathbb{F}_q^{m \times m}$, $\mathcal{S} \in \mathbb{F}_q^{n \times n}$ transformações lineares bijetoras, tais que $\mathcal{P} = \mathcal{T} \circ \mathcal{Q} \circ \mathcal{S}$. Sabendo-se $\mathcal{P}$ e $\mathcal{Q}$, encontre $\mathcal{T}$ e $\mathcal{S}$.*

Em outras palavras, o objetivo do problema IP é encontrar $\mathcal{T}$ e $\mathcal{S}$ para um dado par $(\mathcal{P}, \mathcal{Q})$. Note que, originalmente, $\mathcal{S}$ foi definida como uma transformação afim em vez de linear [Patarin et al. 1998]. Mas, Braeken *et al.* [Braeken et al. 2005, Sec. 3.1] perceberam que a parte constante não é importante para a segurança de certos esquemas $\mathcal{MQ}$ e, portanto, pode ser omitida.

2.3.1. Construção de Chaves $\mathcal{MQ}$

De forma genérica, a chave privada é composta das transformações lineares $\mathcal{T}$ e $\mathcal{S}$ mais a transformação quadrática $\mathcal{Q}$. Note que a transformação $\mathcal{Q}$ possui certa estrutura especial com alçapão (sendo duas estruturas distintas de $\mathcal{Q}$ descritas na Seção 2.3.2 para as assinaturas UOV e Rainbow), que permitirá ao signatário resolver facilmente o sistema $\mathcal{MQ}$ público para gerar assinaturas válidas. A chave pública será dada pela composição $\mathcal{P} = \mathcal{T} \circ \mathcal{Q} \circ \mathcal{S}$. Em alguns esquemas de assinatura não é necessário utilizar transformação $\mathcal{T}$, pois ela se reduz à identidade [Bernstein et al. 2008a, Capítulo 6].

A principal diferença entre esquemas de assinatura $\mathcal{MQ}$ distintos está na estrutura de alçapão de $\mathcal{Q}$. Como a chave pública tem a mesma forma na maioria desses esquemas, o procedimento de verificação de uma assinatura é o mesmo, ou seja, testar se uma dada assinatura $\mathbf{x}$ é solução de um sistema quadrático público $p_k(\mathbf{x}) = h_k(m)$. Para consultar as várias construções distintas do mapa privado $\mathcal{MQ}$ veja [Wolf e Preneel 2005].

Vale mencionar aqui uma otimização óbvia nas matrizes públicas $P^{(k)}$, que proporciona aproximadamente um fator dois de redução. Observando-se a definição do somatório da parte quadrática do polinômio $p_k(x_1, \ldots, x_n)$ (Equação 1), nota-se que o coeficiente do termo $x_i x_j$ é $P^{(k)}_{ij} + P^{(k)}_{ji}$, ou seja, pode-se atualizar o coeficiente $P^{(k)}_{ij}$ da matriz com o valor $P^{(k)}_{ij} + P^{(k)}_{ji}$ e o coeficiente $P^{(k)}_{ji}$ com zero para $i \leq j \leq n$, o que torna cada matriz $P^{(k)}$ triangular superior. Após essa otimização, podemos definir uma única matriz pública que será útil em algumas construções MQ. Trata-se da *matriz pública de coeficientes*, denotada M_P, e é formada linearizando-se os coeficientes de cada uma das m matrizes $P^{(k)}$ já na forma triangular superior. A Figura 2.9, ilustra essa construção.

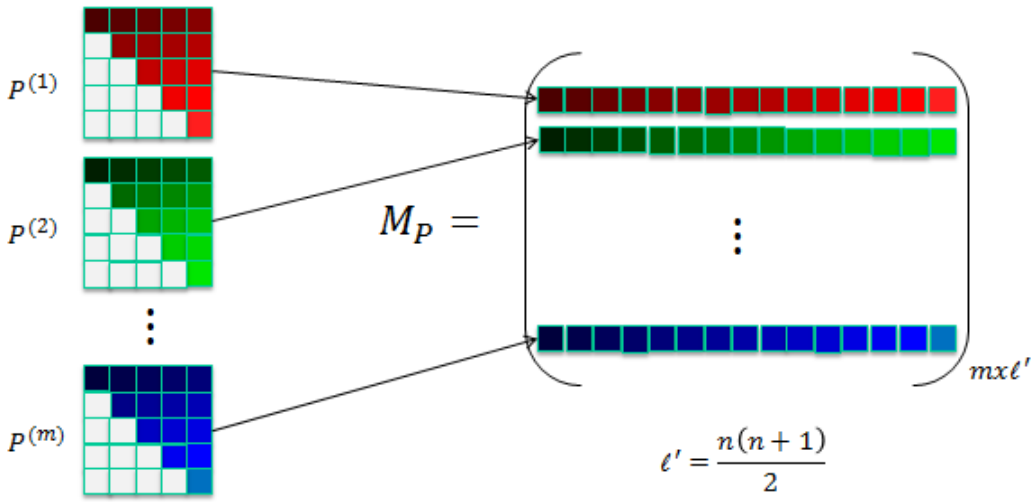


Figura 2.9. Matriz Pública de Coeficientes

2.3.2. Assinaturas UOV e Rainbow

Uma das principais famílias de assinaturas $\mathcal{MQ}$ é a construção *Unbalanced Oil and Vinegar*, ou UOV proposto por Patarin [Kipnis et al. 1999]. O nome UOV provém do fato de as variáveis estarem particionadas em duas classes, chamadas de *vinagres* e *azeites*, de tal maneira que variáveis da classe azeite não ocorrem juntas em nenhum termo

quadrático. Essa estrutura constitui o alçapão, que permite reduzir o sistema quadrático completo a um sistema linear nos azeites, desde que os vinagres sejam conhecidos (ou escolhidos ao acaso), para simplificar os termos onde ocorrem. O alçapão é protegido pela dificuldade conjecturada do problema IP.

Formalmente, o alçapão consiste de um mapa puramente quadrático, denominado mapa central, $Q : \mathbb{F}^n \to \mathbb{F}^m$ com

$$Q = \{f_1(u_1, \ldots, u_n), \ldots, f_m(u_1, \ldots, u_n)\}$$

e

$$f_k(u_1, \ldots, u_n) := \sum_{1 \le i \le j \le n} Q_{ij}^{(k)} u_i u_j \equiv u Q^{(k)} u^{\mathrm{T}} \tag{3}$$

O mapa central tem uma restrição adicional em seus polinômios $f_k(u_1, \ldots, u_n)$. Impõe-se que certa parte de seus coeficientes sejam nulos. O conjunto de variáveis u é dividido em dois subconjuntos, o das variáveis vinagre u_i com $i \in V = \{1, \cdots, v\}$ e o das variáveis azeite u_i com $i \in O = \{v+1, \cdots, n\}$ que contém $m = n - v$ elementos. A restrição é que os polinômios $f^{(k)}$ não contenham nenhum termo cruzando duas variáveis azeite. Isso evita que apareçam termos quadráticos e/ou cruzados nos azeites. Dessa forma, restam apenas termos misturando os seguintes tipos de variáveis $v \times v$ e $o \times v$. Patarin mostrou que, com essa construção, podemos chutar valores arbitrários para os vinagres, resultando em um sistema linear nas variáveis azeite. Esse sistema tem alta probabilidade de apresentar uma solução, i.e. $1 - 1/q$, e pode ser resolvido por eliminação gaussiana com complexidade $\mathcal{O}\left(n^3\right)$. A estrutura dos polinômios privados é a seguinte:

$$f^{(k)}(u_1, \cdots, u_n) := \sum_{i,j \in V,\, i \le j} Q_{ij}^{(k)} u_i u_j + \sum_{i \in V,\, j \in O} Q_{ij}^{(k)} u_i u_j \tag{4}$$

Para gerar uma assinatura $x \in \mathbb{F}_q^n$ de uma dada mensagem, particularmente de seu hash criptográfico $h \in \mathbb{F}_q^m$, o signatário deve inverter a transformação $P(x) = Q(S(x)) = h$. Definindo $x' = xS$, resolve-se primeiro o sistema multivariado, $x' Q^{(k)} x'^{\mathrm{T}} = h_k$, $1 \le k \le m$, para encontrar x'. Por fim, recupera-se a assinatura $x = x' S^{-1}$.

Conforme explicado anteriormente, a estrutura das matrizes $Q^{(k)}$ permite que o sistema $\mathcal{MQ}$ seja resolvido de forma eficiente, escolhendo v variáveis *vinagre* ao acaso e resolvendo o sistema linear resultante para encontrar as m variáveis *azeite* em função dos vinagres escolhidos. Caso o sistema linear não tenha solução, repete-se o processo escolhendo-se novas variáveis vinagre.

Uma assinatura x de h é válida, se e somente, se todos os polinômios $p^{(k)}$ da chave pública forem satisfeitos, i.e. $p^{(k)}(x_1, \cdots, x_n) = P^{(k)}(x) = x P^{(k)} x^T = h_k$, $1 \le k \le m$. A

consistência da verificação $P(x) \stackrel{?}{=} h$ é mostrada a seguir

$$\begin{aligned} P(x) &= xPx^T \\ &= x(Q \circ S)x^{\mathrm{T}} \\ &= x(SQS^T)x^T \\ &= (x'S^{-1})(SQS^{\mathrm{T}})(x'S^{-1})^{\mathrm{T}} \\ &= x'(S^{-1}S)Q(S^{\mathrm{T}}(S^{-1})^T)x'^{\mathrm{T}} \\ &= x'IQIx'^{\mathrm{T}} \\ &= x'Qx'^{\mathrm{T}} \\ &= h. \end{aligned}$$

Historicamente, o esquema UOV deriva da construção *Oil and Vinegar*, ou OV [Patarin 1997], onde o número de vinagres e o número de azeites é o mesmo (balanceado), mas que se mostrou inseguro [Kipnis e Shamir 1998]. Em seguida, notou-se que é possível torna-lo seguro, adotando-se quantidades desbalanceadas de azeites e vinagres $v > m$, o que originou o esquema de assinatura UOV (Unbalanced Oil and Vinegar) [Kipnis et al. 1999]. A Figura 2.10 ilustra a estrutura vetorial de cada polinômio privado UOV

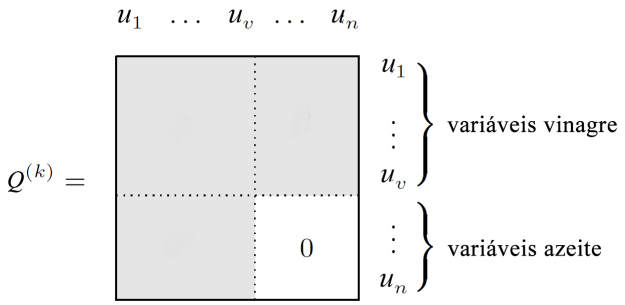


Figura 2.10. Mapa Central UOV

Para camuflar a estrutura dos polinômios $f^{(k)}$, é aplicada uma transformação linear inversível $\mathcal{S} \in \mathbb{F}_q^{nxn}$ à direita de $\mathcal{Q}$. O mapa público resultante é $\mathcal{P} = \mathcal{Q} \circ \mathcal{S}$. A chave privada é dada pelo par $sk := (\mathcal{Q}, \mathcal{S})$ e a chave pública é composta pelos polinômios $pk := P(x_1, \cdots, x_n) = \{p^{(1)}(x_1, \cdots, x_n), \cdots, p^{(m)}(x_1, \cdots, x_n)\}$.

Assim, fica claro que a segurança do sistema não é baseada somente no problema $\mathcal{MQ}$ e, de fato, recuperar a chave privada recai na dificuldade de decompor $\mathcal{P}$ em $\mathcal{Q}$ e $\mathcal{S}$, ou seja, resolver o problema IP.

Uma variante do UOV é a família de assinaturas Rainbow [Ding e Schmidt 2005] proposta por Ding e Schmidt, e tem como principal vantagem a obtenção de assinaturas mais curtas que as encontradas em UOV simples [Thomae 2012, Seção 3].

O Rainbow tem como ideia básica a separação das m equações em bandas e o particionamento das variáveis de acordo, ou seja, cada banda tem seus próprios *vinagres*

e *azeites*, todas as variáveis de uma banda tornam-se os vinagres da banda seguinte, e os azeites de cada banda são calculados em funçãos dos vinagres da mesma.

Em geral o mapa central é dividido em duas bandas, pois essa configuração se mostrou a mais apropriada, no sentido de evitar certos ataques estruturais e ainda assim manter as assinaturas curtas [Thomae 2012].

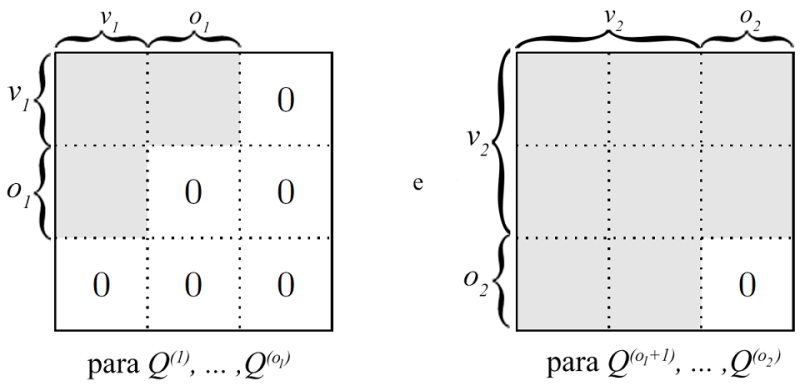


Figura 2.11. mapa central Q do esquema Rainbow de duas bandas

O mapa central Q dessa configuração do Rainbow é dividido em duas camadas caracterizadas pela estrutuda de suas matrizes como na Figura 2.11 onde v_1 e o_1 são o número de vinagres e azeites da primeira camada e v_2 e o_2 são o número de vinagres e azeites da segunda camada, lembrando que v_2 é igual ao número de vinagres e azeites da camada anterior.

O processo de assinatura é semalhante ao UOV, escolhendo ao acaso os *vinagres* da primeira banda para calcular seus *azeites*, da mesma maneira que é feito no UOV, e em seguida usando todas essas variáveis como *vinagres* da próxima banda.

2.3.3. Assinatura Cyclic UOV

Um passo interessante na direção de redução de chaves UOV/Rainbow foi dado através das construções Cyclic UOV/Rainbow [Petzoldt et al. 2010b, Petzoldt et al. 2010a]. Analisando a estrutura das chaves UOV, os autores notaram a existência de uma relação linear entre parte do mapa quadrático público e do mapa quadrático privado. Essa relação pôde ser explorada para gerar pares de chave de forma diferente do usual e reduzir assim o tamanho da chave pública. A ideia foi primeiro calcular a parte quadrática da chave pública com uma estrutura compacta desejada e a partir daí calcular a parte quadrática da chave privada utilizando-se a relação linear encontrada.

Portanto, é possível obter chaves públicas mais compactas enquanto as chaves privadas permanecem com aparência aleatória e sem estrutura. A estrutura sugerida por Petzoldt *et. al.* foi a de matrizes circulantes, daí o nome Cyclic UOV [Petzoldt et al. 2010b]. Matrizes circulantes são bastante compactas, uma vez que podem ser representadas simplesmente por sua primeira linha. Logo, a chave pública pode ser armazenada de forma mais eficiente, além de possibilitar algumas vantagens em processamento, como técnicas de Karatsuba e Fast Fourier Transform (FFT).

A construção das chaves Cyclic UOV pode ser descrita da seguinte maneira. Pri-

meiramente, gera-se uma transformação linear $S \in F_q^{n \times n}$ inversível, onde $S_{ij} \xleftarrow{\$} \mathbb{F}_q, 1 \leq i, j \leq n$ e, a partir de S, calcula-se a relação linear proposta por *Petzoldt et. al.*, denotada por $A_{UOV} := \alpha_{ij}^{rs}$:

$$\alpha_{ij}^{rs} = \begin{cases} S_{ri} \cdot S_{si}, \text{ i=j} \\ S_{ri} \cdot S_{sj} + S_{rj} \cdot S_{si}, \text{ caso contrário} \end{cases}$$

Para ilustrar como as matrizes pública e privada de coeficientes, M_P e M_F, se relacionam, temos inicialmente as Figuras 2.12 e 2.13 que separam as partes de interesse dessas matrizes.

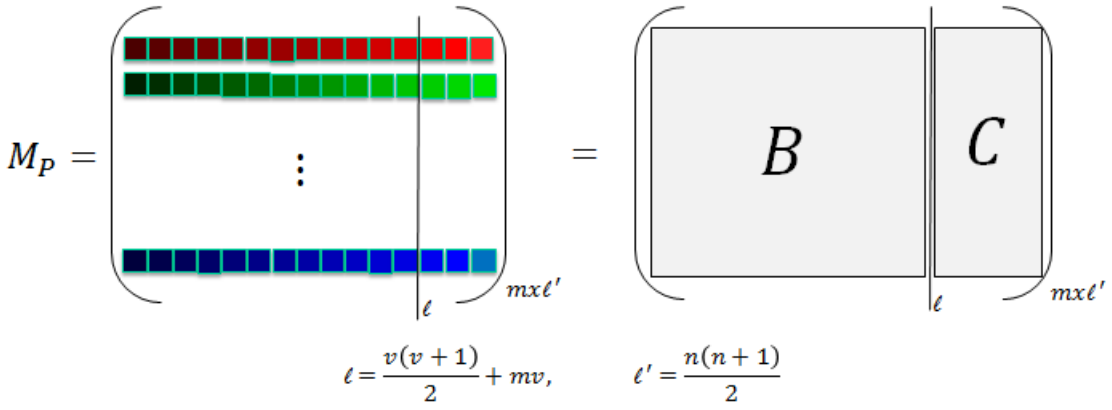


Figura 2.12. Cyclic UOV – Matriz Pública de Coeficientes

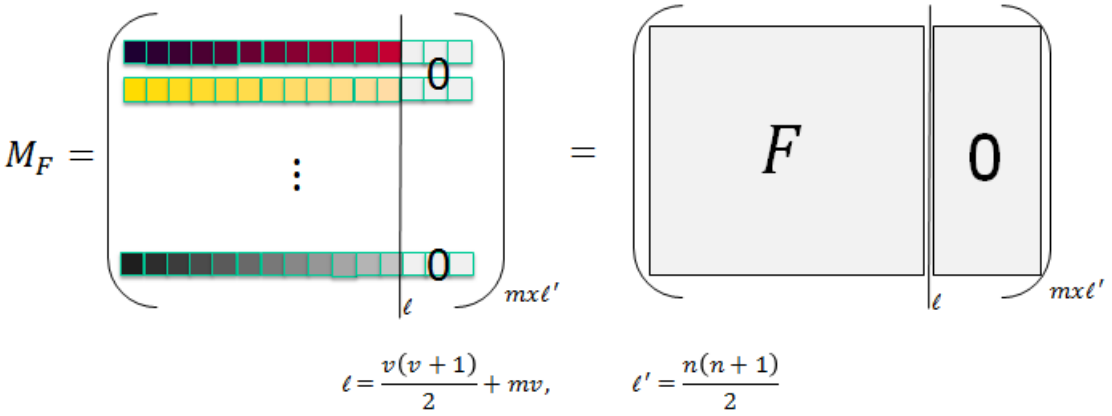


Figura 2.13. Cyclic UOV – Matriz Privada de Coeficientes

Os blocos B de M_P e F de M_F se relacionam através da seguinte expressão $B := F \cdot A_{UOV}(S)$. Dessa forma, na geração de chaves, pode-se primeiro gerar a matriz M_P com B sendo uma matriz circulante e calculando-se, posteriormente, $F := B \cdot A_{UOV}^{-1}(S)$. Essa construção foi capaz de reduzir a chave pública UOV em cerca de 6 vezes para o nível de segurança de 80 bits.

Como mencionado anteriormente, assinaturas $\mathcal{MQ}$ têm se desenvolvido mais intensamente nas duas últimas décadas, reunindo várias propostas em direção à redução

Tabela 2.2. Evolução das assinaturas $\mathcal{MQ}$

proposta	$\|sk\|$	$\|pk\|$	$\|hash\|$	$\|sig\|$	ref.
Rainbow($\mathbb{F}_{2^4}$, 30, 29, 29)	75.8 KiB	113.4 KiB	232	352	[Petzoldt et al. 2010c]
Rainbow($\mathbb{F}_{31}$, 25, 24, 24)	59.0 KiB	77.7 KiB	232	392	[Petzoldt et al. 2010c]
CyclicUOV($\mathbb{F}_{2^8}$, 26, 52)	14.5 KiB	76.1 KiB	208	624	[Petzoldt et al. 2010a]
NC-Rainbow($\mathbb{F}_{2^8}$, 17, 13, 13)	25.5 KiB	66.7 KiB	384	672	[Yasuda et al. 2012]
Rainbow($\mathbb{F}_{2^8}$, 29, 20, 20)	42.0 KiB	58.2 KiB	272	456	[Petzoldt et al. 2010c]
CyclicLRS($\mathbb{F}_{2^8}$, 26, 52)	71.3 KiB	13.6 KiB	208	624	[Petzoldt et al. 2011]
UOVLRS($\mathbb{F}_{2^8}$, 26, 52, 26)	71.3 KiB	11.0 KiB	208	624	[Petzoldt et al. 2011]
CyclicRainbow($\mathbb{F}_{2^8}$, 17, 13, 13)	19.1 KiB	10.2 KiB	208	344	[Petzoldt et al. 2010a]

do tamanho das chaves que é sua principal desvantagem. A Tabela 2.2 ilustra algumas propostas.

2.4. Criptografia baseada em códigos corretores de erros

Nesse capítulo falaremos sobre a teoria e a prática de sistemas criptográficos baseados em códigos corretores de erros.

A teoria da codificação tem como objetivo assegurar que, ao transmitir uma coleção de dados através de um canal sujeito a ruídos (ou seja, a perturbações nos dados enviados), o destinatário dessa transação possa recuperar a mensagem original. Para isso, devem-se encontrar maneiras eficientes de adicionar informação redundante à mensagem original de tal forma que, caso a mensagem chegue ao destinatário contendo erros (existindo inversão em certos bits, para o caso de mensagens binárias), o recebedor possa corrigi-la. A Figura2.14, baseada em [Huffman e Pless 2003], ilustra bem esta situacão:

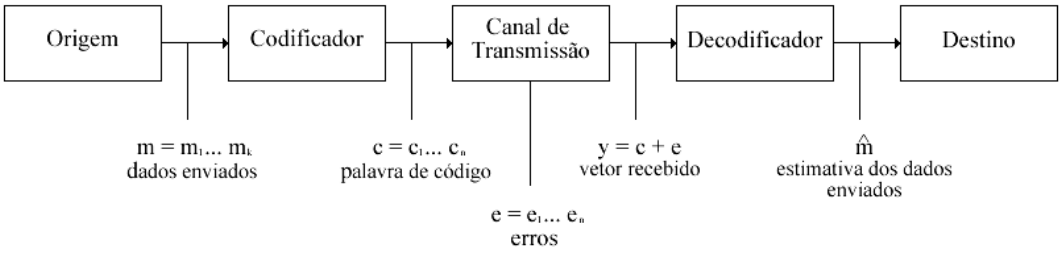


Figura 2.14. Canal de Comunicação

No contexto criptográfico, a primitiva consiste em adicionar erros em uma palavra de um código corretor de erros e calcular uma síndrome relativa a matriz de verificação de paridade desse código.

O primeiro desses sistemas foi um esquema de encriptação de chave pública proposto por Robert J. McEliece em 1978 [McEliece 1978]. A chave privada é um código de Goppa (que será revisado na seção 2.4.1.1) aleatório, binário e irredutível, e a chave pública é uma matriz geradora aleatória com uma versão permutada desse código. O texto encriptado é uma palavra de código no qual alguns erros foram introduzidos, e somente o dono da chave privada pode remover esses erros (e, consequentemente, decriptar a mensagem). Alguns anos mais tarde alguns ajustes de parâmetros foram necessários para

manter o nível de segurança, mas não é conhecido, até hoje, nenhum ataque que possa quebrar esse criptossistema.

2.4.1. Códigos lineares

Para um melhor entendimento técnico desse capítulo, primeiramente vamos explicar algumas noções fundamentais utilizadas no âmbito da criptografia baseada em códigos.

No contexto que segue, todos os índices de matrizes e vetores são numerados a partir de zero, a menos que explicitamente indicado de outra forma. Seja p um primo, e seja $q = p^m$ para algum inteiro $m > 0$. $\mathbb{F}_q$ denota o corpo finito com q elementos. O grau de um polinômio $g \in \mathbb{F}_q[x]$ denota-se $\deg(g)$. Define-se também a noção de *peso de Hamming* e *distância de Hamming*:

Definição 3. *O* peso de Hamming *de um vetor $u \in C \subseteq \mathbb{F}_q^n$ é o número de coordenadas não nulas de u, i.e.* $\mathrm{wt}(u) = \#\{i,\ 0 \leqslant i < n \mid u_i \neq 0\}$. *A* distância de Hamming *entre dois vetores $u, v \in C \subseteq \mathbb{F}_q^n$ é o número* $\mathrm{dist}(u, v)$ *de coordenadas em que esses vetores diferem, i.e.* $\mathrm{dist}(u, v) := \mathrm{wt}(u - v)$.

Agora vamos introduzir alguns conceitos úteis à tarefa de codificar mensagens. O primeiro deles se refere a *código linear*, que pode ser definido como:

Definição 4. *Um código linear $[n, k]$ (de comprimento n e dimensão k) sobre o corpo $\mathbb{F}_q$ é um subespaço vetorial C de dimensão k do espaço $\mathbb{F}_q^n$.*

Um vetor qualquer $u \in C$ é também chamado palavra de código (ou, abreviadamente, uma palavra) de C.

Sendo um espaço vetorial, C é representado por uma base, que pode ser escrita na forma de uma *matriz geradora*:

- Uma matriz geradora G de C é uma matriz sobre $\mathbb{F}_q$ tal que $C = \langle G \rangle$, onde $\langle G \rangle$ indica o espaço vectorial gerado pelas linhas de G. Normalmente as linhas de G são independentes e a matriz possui dimensão $k \times n$. De outra forma: $\exists G \in \mathbb{F}_q^{k \times n} : C = \{uG \in \mathbb{F}_q^n \mid u \in \mathbb{F}_q^k\}$.
- Dizemos que uma matriz geradora G está na forma sistemática se suas primeiras k colunas formam a matriz identidade.
- O chamado *código dual* $C^\perp$ é o código ortogonal de C para o produto escalar sobre $\mathbb{F}_q$ e é um código linear de dimensão $n \times (n - k)$ sobre $\mathbb{F}_q$.

Alternativamente, C é inteiramente caracterizado como o núcleo de uma transformação linear especificada por uma *matriz de verificação de paridade* (ou abreviadamente *matriz de paridade*):

- Uma matriz de paridade H sobre C é uma matriz geradora de $C^\perp$. De outra forma: $\exists H \in \mathbb{F}_q^{r \times n} : C = \{v \in \mathbb{F}_q^n \mid Hv^\intercal = 0^r \in \mathbb{F}_q^r\}$, onde $r = n - k$ é a codimensão de C (ou seja, a dimensão do espaço ortogonal $C^\perp$).

Torna-se fácil ver que G e H, embora não sejam unicamente definidas (pois não há uma única base para C nem para $C^{\perp}$), relacionam-se por $HG^{\mathrm{T}} = O \in \mathbb{F}_q^{r\times k}$.

A transformação linear definida por uma matriz de paridade é chamada *função síndrome* do código. O valor dessa transformação sobre um vetor qualquer $u \in \mathbb{F}_q^n$ denomina-se *síndrome* desse vetor. Claramente, a síndrome de qualquer palavra de código é sempre nula.

Definição 5. *A* distância *(ou* distância mínima*) de um código $C \subseteq \mathbb{F}_q^n$ é a distância de Hamming mínima entre palavras de C, i.e.* $\mathrm{dist}(C) = \min\{\mathrm{wt}(u) \mid u \in C\}$.

Escreve-se $[n,k,d]$ para um código $[n,k]$ cuja distância mínima é (pelo menos) d. Se $d \geqslant 2t+1$, diz-se que o código é capaz de corrigir pelo menos t erros, no sentido de existir não mais que uma única palavra de código a uma distância de Hamming não superior a t de qualquer vetor de $\mathbb{F}_q^n$.

Vários problemas computacionais envolvendo códigos são intratáveis, começando com a própria determinação da distância mínima. Os seguintes problemas têm importância para criptografia baseada em códigos:

Definição 6 (Decodificação geral)**.** *Seja $\mathbb{F}_q$ um corpo finito, e seja (G,w,c) uma tripla consistindo de uma matriz $G \in \mathbb{F}_q^{k\times n}$, um inteiro $w < n$, e um vetor $c \in \mathbb{F}_q^n$. O* problema da decodificação geral (GDP) *consiste em determinar se existe um vetor $m \in \mathbb{F}_q^k$ tal que $e = c - mG$ tenha peso de Hamming* $\mathrm{wt}(e) \leqslant w$.

O problema de busca associado ao GDP consiste em calcular o vetor m dada a palavra com erros c.

Definição 7 (Decodificação de síndromes)**.** *Seja $\mathbb{F}_q$ um corpo finito, e seja (H,w,s) uma tripla consistindo de uma matriz $H \in \mathbb{F}_q^{r\times n}$, um inteiro $w < n$, e um vetor $s \in \mathbb{F}_q^r$. O* problema da decodificação de síndrome (SDP) *consiste em determinar se existe um vetor $e \in \mathbb{F}_q^n$ com peso de Hamming* $\mathrm{wt}(e) \leqslant w$ *tal que $He^{T} = s^{T}$.*

O problema de busca associado ao SDP consiste em calcular o padrão de erro e dada sua síndrome $s_e := eH^{\mathrm{T}}$.

Tanto o problema da decodificação geral quanto o problema da decodificação de síndromes para códigos lineares são *NP*-completos [Berlekamp et al. 1978].

Em contraste com os resultados gerais, o conhecimento da estrutura de certos códigos torna o GDP e o SDP solúveis em tempo polinomial. Uma estratégia básica para definir criptossistemas baseados em códigos é, portanto, manter secreta a informação sobre a estrutura do código, e publicar um código relacionado sem estrutura aparente (portanto, por hipótese difícil de decodificar).

2.4.1.1. Códigos de Goppa

Uma das famílias mais importantes de códigos lineares corretores de erros para fins criptográficos é a dos códigos de Goppa:

Definição 8. *Dado um primo p, $q = p^m$ para algum $m > 0$, uma sequência $L = (L_0, \ldots, L_{n-1}) \in \mathbb{F}_q^n$ de elementos distintos, e um polinômio mônico $g(x) \in \mathbb{F}_q[x]$ de grau t (chamado polinômio gerador) tal que $g(L_i) \neq 0$ para $0 \leqslant i < n$, o* código de Goppa $\Gamma(L, g)$ *é o código $\mathbb{F}_p$-alternante correspondente a* $\mathrm{GRS}_t(L, D)$ *sobre $\mathbb{F}_q$, onde $D = (g(L_0)^{-1}, \ldots, g(L_{n-1})^{-1})$.*

A distância de um código *binário* irredutível de Goppa é pelo menos $2t+1$ [Goppa 1970], e portanto um código de Goppa pode corrigir até t erros (usando, por exemplo, o algoritmo de Patterson [Patterson 1975], às vezes um pouco mais [Bernstein 2011]. Algoritmos de decodificação adequados ainda podem corrigir t erros quando o gerador $g(x)$ não for irredutível mas livre de quadrados. Por exemplo, pode-se equivalentemente ver um código binário de Goppa como o código alternante definido pelo polinômio gerador $g^2(x)$, em cujo caso qualquer decodificador alternante conseguirá decodificar t erros. Os códigos ditos *selvagens* (*wild codes*) estendem esse resultado sob certas circunstâncias [Bernstein et al. 2010]. Para todos os demais casos, não se conhece método de decodificação capaz de corrigir mais que $t/2$ erros.

Definimos, equivalentemente, códigos de Goppa em termos de sua função síndrome:

Definição 9. *Seja $L = (L_0, \ldots, L_{n-1}) \in \mathbb{F}_q^n$ uma sequência (chamada* suporte*) de $n \leqslant q$ elementos distintos, e seja $g \in \mathbb{F}_q[x]$ um polinômio irredutível mônico de grau de t tal que $g(L_i) \neq 0$ para todo i. Para qualquer palavra $e \in \mathbb{F}_p^n$ define-se a* síndrome de Goppa *em forma polinomial $s_e \in \mathbb{F}_q[x]$ como*

$$s_e(x) = \sum_{i=0}^{n-1} \frac{e_i}{x - L_i} \mod g(x). \tag{5}$$

A síndrome é uma função linear de e. Segue daí a seguinte definição alternativa de um código de Goppa:

Definição 10. *O* código de Goppa $[n, n-mt]$ *sobre $\mathbb{F}_p$ com suporte L e polinômio gerador g é o núcleo da função síndrome (Equação 5), i.e. o conjunto de $\Gamma(L, g) := \{e \in \mathbb{F}_p^n \mid s_e \equiv 0 \mod g\}$.*

Escrevendo $s_e(x) := \sum_i s_i x^i$ para algum $s \in \mathbb{F}_q^n$, pode-se mostrar que $s^{\mathrm{T}} = He^{\mathrm{T}}$ com

$$\begin{aligned} H &= \mathrm{toep}(g_1, \ldots, g_t) \\ &\cdot \mathrm{vdm}_t(L_0, \ldots L_{n-1}) \\ &\cdot \mathrm{diag}(g(L_0)^{-1}, \ldots, g(L_{n-1})^{-1}) \end{aligned} \tag{6}$$

Assim, $H = TVD$ onde T é uma matriz de Toeplitz $t \times t$, V é uma matriz de Vandermonde $t \times n$, e D é uma matriz diagonal $n \times n$, de acordo com as seguintes definições:

Definição 11. *Dada uma sequência $(g_1, \ldots, g_t) \in \mathbb{F}_q^t$ para algum $t > 0$, a* matriz de Toeplitz $\mathrm{toep}(g_1, \ldots, g_t)$ *é a matriz $t \times t$ com componentes $T_{ij} := g_{t-i+j}$ para $j \leqslant i$ e $T_{ij} := 0$ nos*

demais casos, ou seja:

$$\text{toep}(g_1,\ldots,g_t) = \begin{bmatrix} g_t & 0 & \ldots & 0 \\ g_{t-1} & g_t & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{bmatrix}.$$

Definição 12. *Dados $t > 0$ e uma sequência $L = (L_0,\ldots,L_{n-1}) \in \mathbb{F}_q^n$ para algum $n > 0$, a* matriz de Vandermonde $\text{vdm}(t,L)$ *é a matriz $t \times n$ com componentes $V_{ij} = L_j^i$, ou seja:*

$$\text{vdm}(t,L) = \begin{bmatrix} 1 & \ldots & 1 \\ L_0 & \ldots & L_{n-1} \\ L_0^2 & \ldots & L_{n-1}^2 \\ \vdots & \ddots & \vdots \\ L_0^{t-1} & \ldots & L_{n-1}^{t-1} \end{bmatrix}.$$

Definição 13. *Dada uma sequência $(d_0,\ldots,d_{n-1}) \in \mathbb{F}_q^n$ para algum $n > 0$, denota-se por* $\text{diag}(d_0,\ldots,d_{n-1})$ *a* matriz diagonal *com componentes $D_{jj} := d_j$, $0 \leqslant j < n$, e $D_{ij} := 0$ nos demais casos, ou seja:*

$$\text{diag}(d_0,\ldots,d_{n-1}) = \begin{bmatrix} d_0 & 0 & \ldots & 0 \\ 0 & d_1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & d_{n-1} \end{bmatrix}.$$

2.4.2. Decodificabilidade

Todos os códigos $[n,k]$ com distância d satisfazem o *limite de Singleton*, que estabelece que $d \leqslant n-k+1$. A existência de um código linear binário $[n,k]$ com distância d é garantida desde que:

$$\sum_{j=0}^{d-2} \binom{n-1}{j} < 2^{n-k}.$$

Este é o chamado *limite Gilbert-Varshamov (GV)*. Códigos binários aleatórios atingem o limite GV, no sentido de que a desigualdade acima aproxima-se muito da igualdade [MacWilliams e Sloane 1977]. Não se conhece nenhuma família de códigos binários, contudo, que possa ser decodificada em tempo subexponencial até o limite GV, nem se conhece algoritmo subexponencial para decodificar códigos gerais até o limite GV.

Considere-se um código $\mathbb{F}_p$-alternante de comprimento n e capaz de corrigir t erros, derivado de um código GRS sobre $\mathbb{F}_{p^m}$. O espaço de síndromes tem tamanho p^{mt}. Contudo, as síndromes decodificáveis são apenas aquelas que correspondem a vetores de erro de peso não superior a t. Em outras palavras, apenas $\sum_{w=1}^{t} \binom{n}{w}(p-1)^w$ síndromes não nulas são univocamente decodificáveis, e assim a sua densidade é

$$\delta = \frac{1}{p^{mt}} \sum_{w=1}^{t} \binom{n}{w}(p-1)^w.$$

Se o comprimento do código é uma fração $1/p^c$ do comprimento máximo para algum $c \geqslant 0$, i.e. $n = p^{m-c}$, a densidade pode ser aproximada por

$$\delta \approx \frac{1}{p^{mt}}\left(\frac{n^t}{t!}\right)(p-1)^t = \frac{(p^{m-c})^t(p-1)^t}{p^{mt}t!} = \left(\frac{p-1}{p^c}\right)^t \frac{1}{t!}.$$

Um caso particularmente bom é, portanto, $\delta \geqslant 1/t!$, que ocorre quando $p^c/(p-1) \leqslant 1$, i.e. $c \leqslant \log_p(p-1)$, ou $n \geqslant p^m/(p-1)$. Infelizmente isso também significa que, para códigos binários, as densidades mais elevadas são atingidas somente por códigos de comprimento máximo ou quase máximo, caso contrário a densidade se reduz de um fator 2^{ct}. Para códigos de comprimento máximo ($n = p^m$ e portanto $c = 0$) a densidade simplifica-se para $\delta \approx (p-1)^t/t!$, que atinge o mínimo relativo $\delta \approx 1/t!$ para códigos binários.

Estaremos interessados também no caso particular de padrões de erro em que uma magnitude particular predomina sobre as demais, e mais especialmente quando todas as magnitudes de erro são iguais. Nesse caso, a densidade de síndromes decodificáveis é $\delta \approx (p-1)/t!$ que novamente atinge o mínimo $\delta \approx 1/t!$ para códigos binários.

2.4.3. Criptossistemas baseados em códigos

Os esquemas originais de encriptação de McEliece [McEliece 1978] e de Niederreiter [Niederreiter 1986], a despeito do nome histórico, mas impreciso e indevido, de *criptossistemas*, são mais bem descritos como *funções de mão única com alçapão* (*trapdoor one-way functions*) do que como métodos completos de encriptação propriamente ditos. Funções dessa natureza podem ser transformadas de diversas maneiras em criptossistemas, por exemplo, através da transformada Fujisaki-Okamoto.

É interessante notar que McEliece e Niederreiter comumente apresentam uma vantagem substancial de velocidade de processamento sobre esquemas mais tradicionais. Por exemplo, sobre um código de comprimento n ambos apresentam complexidade de tempo $O(n^2)$, enquanto os sistemas Diffie-Hellman/DSA, assim como as operações do sistema RSA com expoente privado, têm complexidade de tempo $O(n^3)$ com chaves de n bits.

Por simplicidade, as descrições dos esquemas de McEliece e de Niederreiter a seguir assumem que os padrões de erros corrigíveis são vetores binários de peso t, mas variantes com padrões mais gerais de erro são possíveis, conforme a capacidade de correção do código subjacente. Critérios simples e efetivos para a escolha de parâmetros são oferecidos na seção 2.4.3.3. Cada esquema de encriptação consiste em três algoritmos: GerarChaves, Encriptar e Decriptar.

2.4.3.1. McEliece

- GerarChaves. Dado o nível desejado de segurança λ, escolhem-se um primo p (comumente $p = 2$), um corpo finito $\mathbb{F}_q$ com $q = p^m$ para algum $m > 0$, e um código de Goppa $\Gamma(L, g)$ com suporte $L = (L_0, \ldots, L_{n-1}) \in (\mathbb{F}_q)^n$ (de elementos distintos) e polinômio gerador $g \in \mathbb{F}_q[x]$ de grau t e livre de quadrados, satisfazendo $g(L_j) \neq 0$, $0 \leqslant j < n$. Seja $k = n - mt$. Norteia-se a escolha de modo que o custo de decodificar um código $[n, k, 2t+1]$ seja pelo menos 2^λ passos. Calcula-se uma matriz geradora sistemática $G \in \mathbb{F}_p^{k \times n}$ para $\Gamma(L, g)$, i.e. $G = [I_k \mid -M^T]$ para alguma matriz $M \in \mathbb{F}_p^{mt \times k}$

e sendo I_k a matriz identidade de ordem k. A chave privada é $sk := (L, g)$, e a chave pública é $pk := (M, t)$.

- Encriptar. Para encriptar um texto legível $d \in \mathbb{F}_p^k$, escolhe-se um vetor $e \xleftarrow{\$} \{0,1\}^n \subseteq \mathbb{F}_p^n$ com peso $\mathrm{wt}(e) \leqslant t$, e calcula-se o texto encriptado $c \leftarrow dG + e \in \mathbb{F}_p^n$.

- Decriptar. Para decriptar um texto encriptado $c \in \mathbb{F}_p^n$ com o conhecimento de L e g, calcula-se a síndrome decodificável de c, aplica-se a ela um decodificador para determinar o vetor de erros e, e recupera-se o texto legível d a partir das primeiras k colunas de $c - e$.

2.4.3.2. Niederreiter

- GerarChaves. Dado o nível desejado de segurança λ, escolhem-se um primo p (comumente $p = 2$), um corpo finito $\mathbb{F}_q$ com $q = p^m$ para algum $m > 0$, e um código de Goppa $\Gamma(L, g)$ com suporte $L = (L_0, \ldots, L_{n-1}) \in (\mathbb{F}_q)^n$ (de elementos distintos) e polinômio gerador $g \in \mathbb{F}_q[x]$ de grau t e livre de quadrados, satisfazendo $g(L_j) \neq 0$, $0 \leqslant j < n$. Seja $k = n - mt$. Norteia-se a escolha de modo que o custo de decodificar um código $[n, k, 2t + 1]$ seja pelo menos 2^λ passos. Calcula-se uma matriz de paridade sistemática $H \in \mathbb{F}_p^{mt \times n}$ para $\Gamma(L, g)$, i.e. $H = [M \mid I_{mt}]$ para alguma matriz $M \in \mathbb{F}_p^{mt \times k}$ e sendo I_{mt} a matriz identidade de ordem mt. Finalmente, escolhe-se como parâmetro público uma função de posto de permutação $\phi : \mathcal{B}(n, t) \to \mathbb{Z}/\binom{n}{t}\mathbb{Z}$. A chave privada é $sk := (L, g)$, e a chave pública é $pk := (M, t, \phi)$.

- Encriptar. Para encriptar um texto legível $d \in \mathbb{Z}/\binom{n}{t}\mathbb{Z}$, representa-se d como um padrão de erros $e \leftarrow \phi^{-1}(d) \in \{0,1\}^n \subseteq \mathbb{F}_p^n$ de peso $\mathrm{wt}(e) = t$, e calcula-se como texto encriptado a sua síndrome $s \leftarrow eH^\intercal \in \mathbb{F}_p^{mt}$.

- Decriptar. Para decriptar um texto encriptado $s \in \mathbb{F}_p^{mt}$ com o conhecimento de L e g, transforma-se essa síndrome numa outra decodificável, aplica-se ao resultado um decodificador para determinar o vetor de erros e, e recupera-se a partir deste o texto legível $d \leftarrow \phi(e)$.

2.4.3.3. Parâmetros para criptossistemas baseados em códigos

Os esquemas clássicos de McEliece e Niederreiter, implementados sobre a classe dos códigos de Goppa, permanecem seguros até a data presente, em contraste com implementações sobre muitas outras famílias propostas de códigos [Gibson 1996, Otmani et al. 2010]. De fato, códigos de Goppa têm resistido muito bem a intensas tentativas de criptoanálise, e a despeito do progresso considerável na área [Bernstein et al. 2011] (ver também [Bernstein et al. 2008a] para uma resenha), eles permanecem essencialmente intactos para fins criptográficos desde que foram sugeridos no trabalho pioneiro de McEliece [McEliece 1978].

Tabela 2.3. Parâmetros para McEliece/Niederreiter usando códigos de Goppa binários genéricos

m	n	k	t	lg WF	$\|pk\|$
11	1893	1431	42	80.025	661122
12	2887	2191	58	112.002	1524936
12	3307	2515	66	128.007	1991880
13	5397	4136	97	192.003	5215496
13	7150	5447	131	256.002	9276241

A Tabela 2.3 sugere parâmetros para códigos subjacentes a criptossistemas do tipo McEliece ou Niederreiter e o tamanho $|pk|$ em bits da chave pública resultante. Apenas códigos de Goppa *genéricos* irredutíveis são considerados.

Nota-se que, nesse cenário de códigos genéricos de Goppa, esses esquemas são adversamente afetados por chaves muito grandes em comparação às contrapartidas convencionais. Segue daí a importância de buscar meios de reduzir os tamanhos de chaves, mantendo contudo intacto o nível de segurança associado.

Os primeiros passos rumo ao objetivo de reduzir o tamanho das chaves sem reduzir o nível de segurança em criptossistemas pós-quânticos foram dados por Monico *et al.* através de códigos com (matriz de) verificação de paridade de baixa densidade (*low density parity-check*, ou LDPC) [Monico et al. 2000], depois por Gaborit através de códigos quase-cíclicos [Gaborit 2005], e por Baldi e Chiaraluce através de uma combinação de ambos [Baldi e Chiaraluce 2007].

2.4.4. Códigos LDPC e QC-LDPC

Códigos LDPC foram inventados por Robert Gallager [Gallager 1963] e são códigos lineares obtidos de grafos esparsos bipartidos. Suponha que $\mathscr{G}$ é um grafo com n nós do lado esquerdo (chamados de nós de mensagem) e r nós do lado direito (chamados de nós de verificação), como podemos observar na figura 2.15 abaixo. O grafo da origem a um código linear de bloco de tamanho n e dimensão de ao menos $n-r$ da seguinte forma: as n coordenadas das palavras de códigos são associadas com os n nós de mensagens. As palavras de códigos são os vetores $(c_1,\ldots,c_n)$ de tal modo que, para todos os nós de verificação, a soma das posições vizinhas entre os nós de mensagem seja zero.

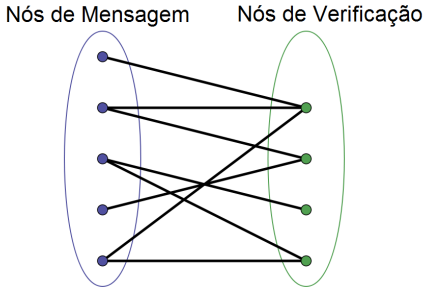


Figura 2.15. Grafo bipartido

A representação por grafos é análoga a representação matricial olhando para a

matriz de adjacência do grafo: seja H uma matriz binária $r \times n$ cuja entrada (i, j) é 1 se e somente se o i-ésimo nó de verificação está conectado ao j-ésimo nó de mensagem no grafo. Então, o código LDPC definido pela grafo é o conjunto de vetores $c = (c_1, \ldots, c_n)$ tal que $H \cdot c^{\mathrm{T}} = 0$. A matriz H é chamada de *matriz de paridade* para o código. Reciprocamente, qualquer matriz binária $r \times n$ da origem a um grafo bipartido entre n nós de mensagens e r nós de verificação, e o código definido como espaço nulo de H é precisamente o código associado a esse grafo. Portanto, qualquer código linear possui uma representação tal como um código associado a um grafo bipartido (note que esse grafo não é definido unicamente pelo código). Entretanto, nem todo código linear binário possui uma representação como um grafo bipartido *esparso*. Se existir, então o código é chamado de código de verificação de paridade de baixa densidade (*low-density parity-check*, ou LDPC).

Uma importante subclasse dos códigos LDPC que apresentam vantagens sobre outros códigos da mesma classe é o dos códigos de verificação de paridade de baixa densidade quase-cíclicos (QC-LDPC) [Tanner 2001]. Em geral, um código QC-LDPC $[n,k]$ satisfaz $n = n_0 b$ e $k = k_0 b$ (e assim também $r = r_0 b$) para algum b, n_0, k_0 (e r_0), e admite uma matriz de paridade que consiste de $n_0 \times r_0$ blocos de submatrizes circulantes esparsas $b \times b$. Um caso particular importante é quando $b = r$ (e também $r_0 = 1$ e $k_0 = n_0 - 1$), uma vez que uma matriz de paridade sistemática para esse código é totalmente definido pela primeira linha de cada bloco $r \times r$. Diz-se que a matriz de paridade está na *forma circulante*.

Contudo, mostrou-se que todas essas propostas contêm vulnerabilidades que as tornam inadequadas para fins criptográficos [Otmani et al. 2010]. Com efeito, o alçapão nesses métodos era protegido essencialmente por nenhum outro mecanismo além de uma permutação privada do código subjacente. A estratégia de ataque nesse panorama consiste em obter um sistema solúvel de equações lineares que os componentes da matriz de permutação precisa satisfazer, e foi montada com sucesso devido à natureza excessivamente restritiva da permutação secreta (uma vez que ela precisa preservar a estrutura quase-cíclica do resultado) e ao fato de que o código secreto é um subcódigo muito particular de um código público.

Uma tentativa de consertar a proposta de Baldi e Chiaraluce chegou a ser proposta [Baldi et al. 2008]. Mais recentemente, Berger *et al*. [Berger et al. 2009] mostraram como evitar os problemas do esquema original de Gaborit e remover as vulnerabilidades até então conhecidas por meio de duas técnicas:

1. Extrair códigos públicos encurtados por blocos de códigos privados muito longos, explorando um teorema devido a Wieschebrink sobre a *NP*-completeza de distinguir códigos encurtados [Wieschebrink 2006];

2. Trabalhar com subcódigos de subcorpo sobre um corpo intermediário entre o corpo e o corpo de extensão do código GRS original adotado pela construção.

Essas técnicas foram aplicadas com algum sucesso a códigos quase-cíclicos. Contudo, a quase totalidade dessa família de códigos foi posteriormente quebrada devido a falhas estruturais de segurança, mais precisamente uma relação entre a estrutura secreta e certos sistemas de equações multivariadas quadráticas [Faugère et al. 2010].

A sabedoria histórica e experimental sugere, portanto, restringir a busca de parâmetros mais eficientes para criptossistemas baseados em códigos à classe dos códigos de Goppa.

2.4.5. Códigos MDPC e QC-MDPC

Uma subclasse interessante em termos de criptografia da família QC-LDPC é dos códigos de verificação de paridade de densidade moderada (MDPC) e sua variante quase-cíclica (QC-MDPC) [Misoczki et al. 2012].

Esses códigos, apresentados por Misoczki *et al.*, possuem densidades próximas o bastante de códigos LDPC que possibilitam a decodificação pelos métodos simples (e indiscutivelmente mais eficientes) de propagação de crença e *bit flipping* de Gallager, ainda denso o bastante para prevenir ataques baseados na presença de palavras muito esparsas no código dual como visto no ataque Stern [Stern 1989] e algumas variantes, sem perder muita capacidade de correção de erros, bem como manter ataques de decodificação baseados em conjuntos de informações [Bernstein et al. 2008b, Bernstein et al. 2011] também inviáveis.

Além disso, para prevenir ataques estruturais como proposto por Faugère *et al.* [Faugère et al. 2010] e por Leander e Gauthier [Umaña e Leander 2010], códigos orientados a criptografia devem se manter o máximo possível sem estrutura, exceto para o alçapão secreto que permite a decodificação privada e, no caso de códigos quase-cíclicos, simetrias externas que permitam uma implementação eficiente. Finalmente, a simetria circulante pode introduzir fraquezas de segurança como apontada por Sendrier [Sendrier 2011], mas isso induz somente um ganho polinomial (especificamente, linear) em eficiência de ataques, e um pequeno ajuste nos parâmetros elimina totalmente esse problema. Densidades típicas nesse caso estão no intervalo de 0.4% a 0.9% do tamanho do código, uma ordem de magnitude acima de códigos LDPC, mas bem abaixo dos publicados em MDPC mencionados acima, e certamente dentro da margem estipulada para códigos de Gallager. A construção é também tão aleatória quanto possível, apenas mantendo a densidade desejada e a geometria circulante, além do tamanho do código ser muito maior que valores típicos para MDPC.

2.4.6. Método de Decodificação de decisão abrupta de Gallager (*Bit Flipping*)

Nessa seção descrevemos o algoritmo de Gallager (*Gallager's hard decision decoding algorithm*, ou mais simplesmente *bit flipping*, seguindo a concisa e clara descrição de Huffman e Pless [Huffman e Pless 2003]. Algoritmo esse necessário para recuperar a mensagem original a partir da palavra de código encriptada com erros.

Assumindo que a palavra de código é encriptada com um código binário LDPC $\mathscr{C}$ para transmissão, e o vetor c é recebido. Para o cálculo da síndrome $s = cH^{\mathrm{T}}$, cada bit recebido de c afeta no máximo d_v componentes dessa síndrome. Se somente o j-ésimo bit de c contiver um erro, então o correspondente d_v de componente s_i de s será igual a 1, indicando as equações de verificação de paridade que não estão satisfeitas. Mesmo se tiver alguns outros bits com erro entre aqueles que contribuíram para o cálculo de s_i, espera-se que vários dos d_v componentes de s serão iguais a 1. Essa é a base do algoritmo de decodificação de Gallager, tanto *hard decision decoding*, quanto *bit-flipping*.

1. Computar cH^{T} e determinar as posições de paridade que estão insatisfeitas (pode-se dizer que essas posições são os componentes de cH^{T} iguais a 1).
2. Para cada um dos n bits, computar o número de posições de paridade insatisfeitas que envolvem aquele bit.
3. Inverta o valor dos bits de c que estão envolvidos no maior número de posições insatisfeitas.
4. Repetir passos 1, 2, e 3 até que se tenha $cH^{\mathrm{T}} = 0$, no caso que c foi decodificado com sucesso, ou até que um certo limite de número de iterações seja alcançado, para o caso que a decodificação do vetor recebido falhou.

O algoritmo de *bit-flipping* não é o melhor método de decodificação para códigos LDPC; de fato, a técnica de propagação de crença [Gallager 1963, Huffman e Pless 2003], é conhecida por exceder sua capacidade de correção de erros. Entretanto, decodificadores de propagação de crença envolvem uma computação com uma *probabilidade* cada vez mais refinada para cada bit da palavra recebida c que contiver um erro, incorrendo em aritmética de ponto flutuante ou aproximações de alta precisão que se adequem ao processo e algoritmos computacionalmente caros. Em um cenário em que o número de erros é fixo e conhecido antecipadamente, como é o caso de aplicações criptográficas, parâmetros podem ser ajustados de tal forma que métodos de decodificação complexos e caros, como a propagação de crença, não são mais necessários.

2.4.7. Assinaturas digitais com códigos corretores de erros

Após algumas tentativas falhas de se criar um esquema de assinatura digital baseada em códigos corretores de erros [Alabbadi e Wicker 1994, Stern 1995], em 2001, Courtois, Finiasz e Sendrier propuseram um esquema promissor [Courtois et al. 2001].

2.4.7.1. CFS

O *CFS* foi proposto como um Sistema de Assinaturas Digitais baseado no Sistema Criptográfico McEliece. Por definição, um sistema de assinatura digital deve prover uma maneira de assinar qualquer documento de tal forma que identifique unicamente seu autor e que disponha de um algoritmo público eficiente de verificação de asssinatura. Para essas tarefas, deve ser escolhido um código linear, ilustrado na explicação como C. Então, o CFS usa uma função de hash pública para resumir o documento m a ser assinado no vetor $h(m)$. Decodificando esse hash com o algoritmo de correção de erros do código escolhido, obtemos um vetor c', correspondendo a assinatura da mensagem m. Para a verificação da assinatura, basta encriptar c', recebido junto da mensagem m, e verificar se corresponde ao cálculo do hash da mensagem m. Como podemos verificar abaixo:

- Geração de Chaves
  1. Escolher um Código de Goppa $G(L, g(X))$
  2. Obter sua matriz $(n-k)\times n$ de verificação de paridade H

3. Calcular $V = SHP$, onde S é uma matriz binária inversível $(nk)\times(nk)$ aleatória e P uma matriz de permutação aleatória $n \times n$.

Assim, as chaves seriam: Chave Privada = G, Chave Pública = (V, t).

- Assinatura

1. Encontrar o menor $i \in \mathbb{N}$ tal que, para $c = h(m, i)$ e $c' = S^{-1}c$, c' seja uma síndrome decodificável de G.
2. Usando o algoritmo de decodificação de G, obter o vetor de erros e', cuja síndrome seja c', ou seja $c' = H(e')^t$.
3. Obter $e^t = P^{-1}(e')^t$.

Assim, a assinatura é o par: (e, i).

- Verificação de Assinatura

1. Obter $c = Ve^t$.
2. Aceite somente se $c = h(m, i)$

Apesar do CFS ser um esquema de assinatura ainda seguro após passar por várias criptoanálises, não é adequado para aplicações padrão comumente utilizadas hoje, já que além do tamanho das chaves públicas o custo para assinar são muito grandes para um conjunto de parâmetros seguros.

2.5. Criptografia baseada em reticulados

O uso de reticulados em criptografia surgiu com o trabalho de Ajtai [Ajtai 1996], onde o problema de encontrar vetores pequenos em uma classe aleatória de reticulados é utilizado como base da demonstração da existência de funções de mão única. A criptografia baseada em reticulados, além de estar amparada por demonstrações de segurança construídas sobre a dificuldade do pior caso de determinados problemas, possui implementações eficientes e descrições relativamente simples de serem compreendidas.

A criptografia baseada em reticulados pode ser dividida em duas categorias: (i) com demonstração de segurança, como por exemplo o esquema de Ajtai ou aqueles com base no problema LWE, cujos algoritmos envolvem operações com uma matriz A, que, como veremos adiante, está associada a chave pública, fazendo com que a encriptação e decriptação tenham complexidade quadrática ou mesmo cúbica, deixando a desejar em relação a criptografia convencional; ou (ii) sem demonstração de segurança, porém com implementação eficiente, como o NTRU. Um desafio recentemente resolvido foi o de prover demonstração de segurança, com base no pior caso de problemas em reticulados ideais, para esquemas com performance semelhante ao NTRU [Stehlé e Steinfeld 2011]. Embora problemas intratáveis em reticulados possam não ser intratáveis em reticulados ideais, não é conhecido nenhum algoritmo polinomial para resolvê-los, mesmo com fator de aproximação polinomial ou com o uso de computadores quânticos.

2.5.1. Fundamentos

Definição 14. *Formalmente, reticulados são definidos como a combinação linear de n elementos $b_1, ..., b_n \in \mathbb{R}^m$, linearmente independentes, denominados base do reticulado.*

$$\mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \right\}.$$

Em outras palavras, um reticulado é um espaço vetorial discretizado, ou seja, existe uma analogia que nos permite utilizar conceitos como norma, dimensão, ortogonalidade, transformação linear, entre outros. Uma maneira alternativa de abordar o assunto é por meio de notação matricial, onde a base pode ser representada por uma matriz $B = [b_1, ..., b_n]$, pertencente a $\mathbb{R}^{m \times n}$. O reticulado gerado pela matriz B é definido por $\mathcal{L} = \{Bx \mid x \in \mathbb{Z}^n\}$, de forma que o determinante $\det(B)$ é independente da escolha da base e corresponde geometricamente ao inverso da densidade de pontos do reticulado em $\mathbb{Z}^m$.

Definição 15. *Dado um reticulado $\mathcal{L}(B)$, os vetores que constituem a base do reticulado podem ser encarados como arestas de um paralelepípedo de dimensão n. Assim, podemos definir $\mathcal{P}(B) = \{Bx \mid x \in [0, 1)^n\}$, denominado paralelepípedo fundamental de B. Podemos redefinir o paralelepípedo de forma a obter uma região simétrica. Para isso, seja $\mathcal{P}_{1/2}(B) = \{Bx \mid x \in [-1/2, 1/2)^n\}$, denominado paralelepípedo fundamental centralizado de B. As figuras 2.16 e 2.17 ilustram exemplos de paralelepípedos fundamentais (centralizado ou não) em dimensão 2.*

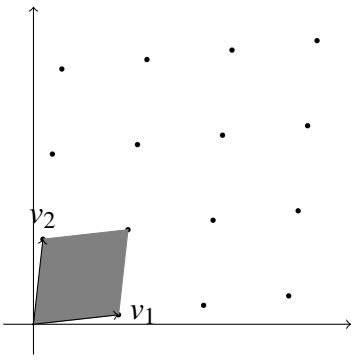


Figura 2.16. $\mathcal{P}(B)$

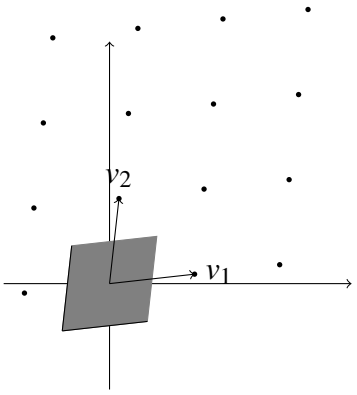


Figura 2.17. $\mathcal{P}_{1/2}(B)$

Teorema 1. *Seja $\mathcal{L} \in \mathbb{R}^m$ um reticulado de dimensão n e seja $\mathcal{F}$ o paralelepípedo fundamental de $\mathcal{L}$, então dado um elemento $w \in \mathbb{R}^m$, podemos escrever w na forma $w = v + t$, para $v \in \mathcal{L}$ e $t \in \mathcal{F}$ únicos. Esta equação pode ser encarada como uma redução modular, onde o vetor t é interpretado como w* $(\mathrm{mod}\ \mathcal{F})$.

O volume do paralelepípedo fundamental é dado por $\mathrm{Vol}(\mathcal{F}) = |\det(B)|$. Dadas duas bases $B = \{b_1, ..., b_n\}$ e $B' = \{b'_1, ..., b'_n\}$ de um mesmo reticulado $\mathcal{L}$, temos que $\det(B) = \pm \det(B')$.

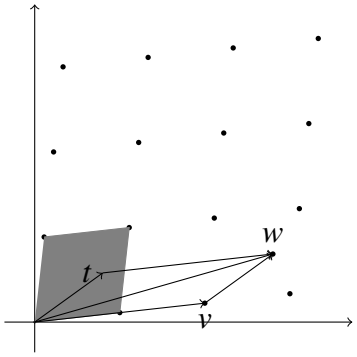


Figura 2.18. Redução módulo $\mathcal{P}(B)$

O problema de encontrar o vetor de norma mínima (*shortest vector problem* - SVP) é uma das questões fundamentais em reticulados. Rigorosamente, dado o reticulado $\mathcal{L}(B)$, deseja-se encontrar o vetor não nulo com norma mínima. Na prática, é utilizado um fator de aproximação $\gamma(n)$ para o problema SVP, isto é, deseja-se encontrar um vetor cuja norma seja inferior ao vetor de norma mínima, multiplicada por $\gamma(n)$.

Outros problemas em reticulados, importantes do ponto de vista da criptografia, são:

- o *problema do vetor de distância mínima* (*closest vector problem* - CVP). Dados um reticulado $\mathcal{L}(B)$ e um vetor $t \in \mathbb{R}^m$, o objetivo é encontrar o vetor $v \in \mathcal{L}(B)$ que seja mais próximo de t;
- e o *problema dos vetores independentes mínimos* (*shortest independent vector problem* - SIVP). Dada uma base $B \in \mathbb{Z}^{m \times n}$, o problema consiste em encontrar n vetores linearmente independentes $(v_1, ..., v_n)$, pertencentes ao reticulado, tais que a norma máxima entre os vetores v_i seja mínima.

Definição 16. *Dado um reticulado $\mathcal{L}$ e uma base $B = (v_1, ...v_n)$, a razão de Hadamard, denotada por $\mathcal{H}(B)$, é definida da seguinte maneira:*

$$\mathcal{H}(B) = \left(\frac{\det \mathcal{L}}{\prod_{1 \leq i \leq n} \|v_i\|} \right)^{1/n}.$$

É fácil mostrar que, para qualquer base B, temos que $0 \leq \mathcal{H}(B) \leq 1$. Além disso, quanto mais próximo de 1 mais ortonormal é a base.

Uma classe particularmente importante de reticulados é aquela formada pelos *reticulados q-ários*, denotados por Λ_q. Dado um inteiro q, os vetores do reticulado são restritos, coordenada a coordenada, a elementos de $\mathbb{Z}_q$. Dada uma matriz $A \in \mathbb{Z}_q^{n \times m}$, o reticulado q-ário é determinado pelas linhas de A, ao invés das colunas. Ou seja, é formado pelos vetores $y = A^T s \pmod q$, para $s \in \mathbb{Z}^n$. O reticulado q-ário ortogonal, $\Lambda_q^{\perp}$, em relação a matriz A, é dado pelos vetores y tais que $Ay = 0 \pmod q$. Dado um reticulado

$\mathcal{L}$, o *reticulado dual*, $\mathcal{L}^*$, é formado pelos vetores y, tais que $\langle x, y\rangle \in \mathbb{Z}$, para $x \in \mathcal{L}$. Em especial, o reticulado q-ário ortogonal, $\Lambda_q^{\perp}(A)$, é igual a $q\Lambda_q(A)^*$.

2.5.1.1. Algoritmo LLL

Nesta seção será descrito o algoritmo LLL, que calcula uma nova base para um determinado reticulado, com razão de Hadamard mais próxima a 1. Isto é, o algoritmo LLL realiza o que chamamos de redução de base, porque os vetores da nova base tem maior ortonormalidade que a base original. Portanto, este algoritmo pode ser utilizado para resolver o problema SVP, como veremos adiante.

Em um espaço vetorial com base $(v_1, \cdots, v_n)$, a obtenção de uma base ortonormal é realizada pelo algoritmo Gram-Schmidt. A ideia da redução de Gauss é a mesma que está por trás do algoritmo Gram-Schmidt, onde temos que $\mu_{ij} = v_i v_j^*/\|v_j^*\|^2$, mas os valores de μ_{ij} não são necessariamente inteiros, de modo que a redução de Gauss considera o uso do inteiro mais próximo $\lfloor \mu_{ij} \rceil$. O algoritmo termina quando este inteiro mais próximo é zero, condição que apenas em dimensão 2 garante que o menor vetor foi encontrado.

Algoritmo 2.5.1 Redução de Gauss

Entrada: Uma base (v_1, v_2).

Saída: Retorna uma base com o menor vetor do reticulado (v_1^*) e com um vetor v_2^* que não pode ser reduzido pela subtração de v_1.

$v_1^* = v_1$ e $v_2^* = v_2$.
enquanto verdade **faça**
  se $\|v_2^*\| < \|v_1^*\|$ **então**
    troque v_1^* com v_2^*
  Calcule $m = \lfloor v_1^*.v_2^*/\|v_1^*\|^2 \rceil$.
  se $m = 0$ **então retorne** (v_1^*, v_2^*).
  Troque v_2^* por $v_2^* - mv_1^*$.

Definição 17. *Seja $B = (v_1, ..., v_n)$ a base de um reticulado $\mathcal{L}$ e seja $B^* = (v_1^*, ..., v_n^*)$ a base retornada pelo algoritmo Gram-schmidt. A base B é denominada LLL reduzida se forem satisfeitas as seguintes condições:*

Condição de norma: $|\mu_{i,j}| = \frac{v_i.v_j^*}{\|v_j^*\|^2} \leq \frac{1}{2}$ *para todo* $1 \leq j < i \leq n$.

Condição de Lovász: $\|v_i^*\|^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2)\|v_{i-1}^*\|^2$ *para todo* $1 < i \leq n$.

Teorema 2. *Seja B uma base LLL reduzida de um reticulado $\mathcal{L}$, então B resolve o problema SVP com fator de aproximação $2^{(n-1)/2}$.*

Um ponto importante que merece destaque é a escolha do valor 3/4. Se este valor fosse substituído por 1, obteríamos a redução de Gauss. Porém, não se sabe se o algoritmo iria terminar em tempo polinomial. Na verdade, qualquer valor estritamente menor que 1 faz com que o algoritmo termine em tempo polinomial. Com isso, criptossistemas

Algoritmo 2.5.2 LLL

Entrada: Uma base $(v_1, ..., v_n)$.

Saída: Retorna uma base com o menor vetor do reticulado (v_1^*) e com um vetor v_2^* que não pode ser reduzido pela subtração de v_1.

$k = 2$.
$v_1^* = v_1$.
enquanto $k \leq n$ **faça**
  para $j = 1$ até $j = k - 1$ **faça**
    $v_k = v_k - \lfloor \mu_{k,j} \rceil v_j^*$.
  se $\|v_k^*\|^2 \geq (\frac{3}{4} - \mu_{k,k-1}^2)\|v_{k-1}^*\|$ **então**
    $k = k + 1$.
  senão
    Troque v_{k-1} com v_k.
  retorne $(v_1, ..., v_n)$.

baseados em problemas como SVP e CVP devem ter seus parâmetros ajustados para evitar ataques que utilizam o algoritmo LLL.

Em linhas gerais, dada uma base $(v_1, ..., v_n)$, é possível obter uma nova base satisfazendo a condição de tamanho, simplesmente subtraindo de v_k múltiplos de $v_1, ..., v_{k-1}$ de modo a reduzir o valor absoluto de v_k. Se a condição de norma for satisfeita, verifica-se se a condição de Lovász também é satisfeita, caso não seja, os vetores são reordenados e novamente realiza-se a redução de norma.

2.5.2. Hash baseado em reticulados

O primeiro criptossistema baseado em reticulados foi proposto por Ajtai [Ajtai 1996]. Este trabalho tem grande importância porque a demonstração de segurança foi realizada com base no pior caso de problemas em reticulados. Isto é, inverter a função de hash em média tem a mesma dificuldade que o pior caso do problema SVP em reticulados q-ários duais.

Especificamente, dados os inteiros n, m, d, q, é construída uma família de hash criptográficos, $f_A : \{0, ..., d-1\}^m \to \mathbb{Z}_q^n$, indexada pela matriz $A \in \mathbb{Z}_q^{n \times m}$. Dado um vetor y, temos que $f_A(y) = Ay \pmod q$. O algoritmo 2.5.3 detalha essas operações. Uma escolha possível para os parâmetros é $d = 2, q = n^2, m \approx 2n \log q / \log d$, de modo a obter um fator de compressão de 2.

A segurança do esquema está no fato de que encontrar uma colisão $f_A(y) = f_A(y')$, implica na existência de um vetor pequeno, $y - y'$, no reticulado $\mathcal{L}_q^*(A)$.

Algoritmo 2.5.3 Hash de Ajtai

Entrada: Inteiros $n, m, q, d \geq 1$. Uma matriz A escolhida uniformemente em $\mathbb{Z}_q^{n \times m}$. Um vetor $y \in \{0, ..., d-1\}^m$.

Saída: Um vetor $f(y) \in \mathbb{Z}_q^n$.

retorne $f(y) = A.y \pmod q$.

Esta proposta é bastante simples e pode ser implementada eficientemente, porém, na prática, as funções de hash são projetadas de forma *ad-hoc*, sem as garantias fornecidas por uma demonstração de segurança, sendo assim mais eficientes que a construção de Ajtai. Além disso, com uma quantidade suficientemente grande de valores da função, um adversário consegue reconstruir o paralelepípedo fundamental do reticulado $\mathcal{L}_q^*(A)$, permitindo encontrar colisões facilmente. Em 2011, Stehlé e Steinfeld [Stehlé e Steinfeld 2011] propuseram uma família mais eficiente de funções de hash resistentes a colisão, cuja construção será importante para esquemas de assinatura digital, como veremos adiante.

2.5.3. Encriptação baseada em reticulados

2.5.3.1. GGH

O criptossistema GGH [Goldreich et al. 1997] permite compreender facilmente o uso de reticulados no contexto de criptografia de chave pública. Este criptossistema utiliza o conceito de ortonormalidade da base na definição do par de chaves. A chave privada é definida como uma base B_{priv} do reticulado, formada por vetores quase ortogonais e com norma pequena.

De modo geral, o criptossistema GGH funciona da seguinte maneira:

- o algoritmo de encriptação acrescenta o ruído $r \in \mathbb{R}^n$ ao texto claro $m \in \mathcal{L}$, gerando o texto encriptado $c = m + r$;
- o algoritmo de decriptação precisa ser capaz de retirar o ruído inserido. Alternativamente, é preciso resolver uma instância do problema CVP.

A figura 2.19 mostra um reticulado em dimensão 2, com base dada pelos vetores v_1 e v_2, praticamente ortogonais. Já a figura 2.20 mostra uma base para o mesmo reticulado, composta por vetores pouco ortogonais.

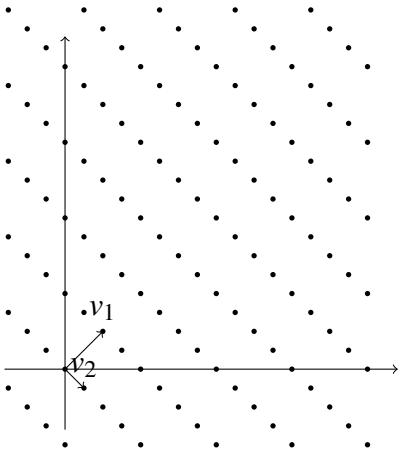


Figura 2.19. Base boa

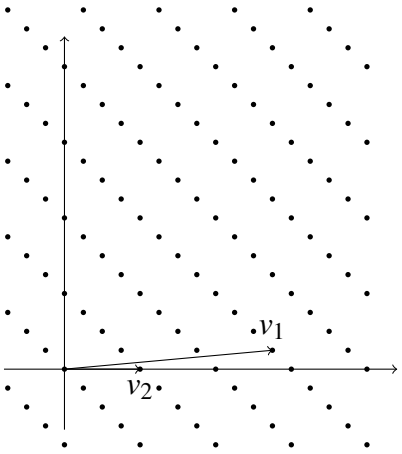


Figura 2.20. Base ruim

Conforme menor a ortonormalidade da base conhecida e maior a dimensão do reticulado, mais difícil é o problema CVP. Desta forma, a chave pública pode ser definida

por uma base B_{pub} do reticulado, tal que $\mathcal{H}(B_{\text{pub}})$ seja aproximadamente zero. Por outro lado, com o conhecimento da chave privada B_{priv}, o algoritmo de Babai [Babai 1986], definido a seguir, pode ser utilizado para recuperar o texto claro.

A ideia geral do algoritmo de Babai é representar o vetor c na base privada B_{priv}, resolvendo um sistema de n equações lineares. Como $c \in \mathbb{R}^{n \times n}$, para obter um elemento do reticulado $\mathcal{L}$, cada coeficiente $t_i \in \mathbb{R}^n$ é aproximado para o inteiro mais próximo a_i, onde esta operação de arredondamento é denotada por $a_i \leftarrow \lfloor t_i \rceil$. Este procedimento simples funciona bem desde que a base B_{priv} seja suficientemente ortonormal, reduzindo os erros do arredondamento.

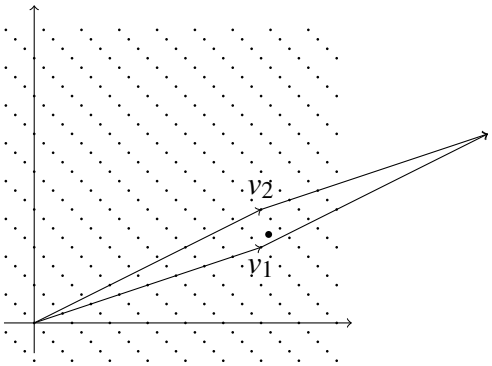


Figura 2.21. CVP com base ruim

Uma forma de atacar o criptossistema GGH é tentar melhorar a base B_{pub}, ou seja, tornar seus vetores menores e mais ortogonais. Em dimensão 2 o problema pode ser facilmente resolvido usando o algoritmo de Gauss (algoritmo 2.5.1). Para dimensões grandes o problema é considerado difícil, mas em 1982 houve grande avanço, com a publicação do algoritmo LLL [Lenstra et al. 1982]. Sendo assim, os parâmetros do criptossistema devem ser projetados para que o algoritmo LLL não possa ser usado para resolver o problema CVP.

2.5.3.2. NTRU

O criptossistema NTRU [Hoffstein et al. 1998] é construído sobre aneis polinomiais, mas também pode ser definido com base em reticulados, isto porque o problema

Algoritmo 2.5.4 Algoritmo de Babai

Entrada: o reticulado $\mathcal{L}$ de dimensão n; o vetor $c_{B_{\text{pub}}} = (c_1, ..., c_n)$, onde $c_i \in \mathbb{R}$; e uma base $B_{\text{priv}} = (s_1, ..., s_n)$, suficientemente ortonormal.

Saída: o vetor $m \in \mathcal{L}$ que resolve o problema CVP com relação a c e $\mathcal{L}$.

Resolva um sistema de n equações, $c = t_1 s_1 + ... + t_n s_n$, nas variavéis t_i, onde $1 \leq i \leq n$.

para $i = 0$ até $i = n$ **faça**

$a_i \leftarrow \lfloor t_i \rceil$

retorne $m \leftarrow a_1 s_1 + ... a_n s_n$

subjacente pode ser interpretado como sendo o SVP e o CVP. Desta maneira, a solução de algum desses problemas representaria um ataque ao criptossistema, de modo que os parâmetros devem ser ajustados para que o algoritmo LLL não possa se usado para este fim.

O criptossistema utiliza os seguintes aneis polinomiais: $R = \mathbb{Z}[x]/(x^N - 1)$, $R_p = (\mathbb{Z}/p\mathbb{Z})[x]/(x^N - 1)$ e $R_q = (\mathbb{Z}/q\mathbb{Z})[x]/(x^N - 1)$, onde N, p, q são inteiros positivos.

Definição 18. *Dados os inteiros positivos d_1 e d_2, define-se $\mathcal{T}(d_1, d_2)$ como sendo a classe de polinômios tais que existam d_1 coeficientes iguais a 1, d_2 coeficientes iguais a -1 e o restante dos coeficientes iguais a zero. Estes polinômios são denominados* polinômios ternários.

Os parâmetros públicos são dados por (N, p, q, d), onde N e p são primos, $(p, q) = (N, q) = 1$ e $q > (6d + 1)p$. A chave privada corresponde aos polinômios $f(x) \in \mathcal{T}(d + 1, d)$ e $g(x) \in \mathcal{T}(d, d)$. A chave pública é o polinômio $h(x) \equiv F_q(x).g(x)$, onde $F_q(x)$ é o inverso multiplicativo de $f(x)$ em R_q.

Dada uma mensagem $m(x) \in R$, cujos coeficientes estejam no intervalo entre $-p/2$ e $p/2$, escolhe-se aleatoriamente $r(x)$ e calcula-se o texto encriptado $e(x) \equiv ph(x).r(x) + m(x) \pmod{q}$.

Para decriptar, primeiramente calcula-se a função $a(x) \equiv f(x).e(x) \pmod{q}$, de tal modo que os coeficientes estejam entre $-q/2$ e $q/2$, para então obter novamente a mensagem $m(x)$, tal que $m(x) \equiv F_p(x).a(x) \pmod{p}$.

- GerarChaves. Escolhe-se $f \in \mathcal{T}(d + 1, d)$ tal que f possua inversa em R_q R_p. Escolhe-se também $g \in \mathcal{T}(d, d)$. CalculeF_q como sendo o elemento inverso de f em R_q e, analogamente, F_p o elemento inverso de f em R_p. A chave pública é dada por $h = F_q.g$.

- Encriptar. Dado o texto claro $m \in R_p$, escolhe-se aleatoriamente $r \in \mathcal{T}(d, d)$ e calcule $e \equiv pr.h + m \pmod{q}$, onde h é a chave pública do destinatário.

- Decriptar. Calcule $a = \lfloor f.e \equiv pg.r + f.m \rceil_q$. Finalmente, a mensagem pode ser obtida pelo cálculo $m \equiv F_p.a \pmod{p}$.

2.5.3.3. Encriptação baseada no problema LWE (Learning with Errors)

Assim como o GGH, o NTRU não possui demonstração de segurança. Nesta seção será apresentado o criptossistema baseado no problema LWE, que é uma proposta eficiente e que possui demonstração de segurança com base no pior caso de problemas em reticulados [Regev 2010]. Esta demonstração é uma redução quântica, isto é, mostra que uma vulnerabilidade do criptossistema implica em um algoritmo quântico para resolver problemas em reticulados. Em 2009, Peikert mostrou uma redução clássica na demonstração de segurança do problema LWE [Peikert 2009].

Definição 19. *O problema LWE consiste em encontrar o vetor $s \in \mathbb{Z}_q^n$, dadas as equações $\langle s, a_i \rangle + e_i = b_i \pmod{q}$, para $1 \leq i \leq n$. Os valores e_i são pequenos erros que são inseridos, de acordo com uma distribuição $\mathcal{D}$, geralmente tomada como sendo a distribuição normal.*

Em 2010, Lyubaskevsky, Peikert e Regev usaram aneis polinomiais para definir o esquema RLWE [Lyubashevsky et al. 2010]. Seja $f(x) = x^d + 1$, onde d é uma potência de 2. Dado um inteiro q e um elemento $s \in R_q = \mathbb{Z}_q[x]/f(x)$, o *problema LWE em anel* sobre R_q, com relação a uma distribuição $\mathcal{D}$, é definido equivalentemente, ou seja, é preciso encontrar s que satisfazendo as equações $s.a_i + e_i = b_i \pmod{R_q}$, para $1 \leq i \leq n$, onde a_i e b_i são elementos de R_q e a redução modular em R_q é o mesmo que reduzir o polinômio resultante módulo $f(x)$ e seus coeficientes módulo q. Assim, o criptossistema baseado no problema LWE pode ser construído da seguinte maneira:

- GerarChaves. Escolhe-se aleatoriamente $a \in R_q$ e utiliza-se a distribuição $\mathcal{D}$ para gerar s e e em R. A chave priva é dada por s, enquanto a chave pública é dada por $(a, b = a.s + e)$.
- Encriptar. Para encriptar d bits, é possível interpretar esses bits como coeficientes de um polinômio em R. O algoritmo de encriptação então escolhe $r, e_1, e_2 \in R$, usando a mesma distribuição $\mathcal{D}$ e calcula (u, v) da seguinte maneira:

$$\begin{aligned} u &= a.r + e_1 \pmod{q}, \\ v &= b.r + e_2 + \lfloor q/2 \rfloor .z \pmod{q}. \end{aligned}$$

- Decriptar. Por sua vez, o algoritmo de decriptação calcula

$$v - u.s = (r.e - s.e_1 + e_2) + \lfloor q/2 \rfloor .z \pmod{q}.$$

De acordo com a escolha de parâmetros, temos que $(r.e - s.e_1 + e_2)$ tem tamanho no máximo $q/4$, de modo que os bits do texto claro podem ser calculados verificando cada coeficiente do resultado obtido. Se o coeficiente for mais próximo de 0 que de $q/2$, então o bit correspondente é 0, caso contrário é 1.

Alguns conceitos desta seção, como por exemplo o uso do polinômio ciclotômico $f(x)$ e a distribuição gaussiana $\mathcal{D}$, foram recentemente incorporados ao esquema NTRU, permitindo com isso obter um esquema semanticamente seguro e eficiente para encriptação baseada em reticulados [Stehlé e Steinfeld 2011], cujas chaves pública e privada e algoritmos de encriptação e decriptação têm complexidade $\tilde{O}(\lambda)$, onde λ é o parâmetro de segurança.

2.5.3.4. Encriptação homomórfica

Recentemente, Gentry propôs a construção de um esquema de encriptação completamente homomórfica [Gentry 2009], resolvendo assim um problema que estava em

aberto desde 1978, quando Rivest, Adleman e Dertouzos conjecturaram a existência de homomorfismos secretos [Morais e Dahab 2012], onde a função de encriptação também é um homomorfismo algébrico. Em outras palavras, é possível somar e multiplicar textos encriptados, de modo que, ao decriptá-los, obtêm-se o resultados das mesmas operações, realizadas com o texto claro correspondente.

Se o espaço de texto claro for dado por $\{0, 1\}$, então a soma de bits é equivalente a uma disjunção exclusiva lógica, enquanto a multiplicação é equivalente a uma conjunção lógica. Portanto, é possível computar qualquer circuito booleano sobre dados encriptados, o que permite a construção de máquinas de Turing, sendo assim possível executar homomorficamente qualquer algoritmo com argumentos encriptados, gerando uma saída encriptada.

Utilizando a encriptação homomórfica é possível delegar a computação de algoritmos a um servidor, mantendo o sigilo dos dados de entrada. Isto é interessante para a computação em nuvem, permitindo a construção de aplicações como por exemplo banco de dados encriptado, encriptação de disco, mecanismos de busca sobre consultas encriptadas, etc.

Brakerski propôs o uso do problema LWE para construção de encriptação completamente homomórfica [Brakerski e Vaikuntanathan 2011], reduzindo a complexidade dos algoritmos, realizando cada operação homomórfica em tempo polilogarítmico. Brakerski utilizou uma nova maneira para gerenciar o crescimento do ruído, tornando possível a realização de uma quantidade maior de multiplicações. Em especial, ele propôs um algoritmo para redução de módulo, implicitamente reduzindo a taxa de crecimento do ruído. Outro algoritmo proposto, a redução de dimensão, permitiu substituir o algoritmo de autoinicialização por um novo método, com parâmetros públicos menores. Todavia, mesmo considerando as otimizações propostas recentemente, a encriptação completamente homomórfica ainda não é prática o suficiente.

2.5.4. Assinatura digital

Os criptossistemas GGH e NTRU podem ser transformados para construir esquemas de assinatura digital [Bernstein et al. 2008a]. Todavia, tais propostas não possuem demonstração de segurança e, de fato, existem ataques que permitem recuperar a chave privada dada uma quantidade suficientemente grande de pares de mensagem e assinatura [Nguyen e Regev 2006].

Em 2007, Gentry, Peikert e Vaikuntanathan [Gentry et al. 2008] criaram um novo tipo de função alçapão, f, com uma propriedade extra: um algoritmo eficiente que, com auxílio do alçapão, retorna elementos da pré-imagem de f (*preimage sampling*). Para isso, é usada uma composição de distribuições normais para obter um ponto próximo a um elemento do reticulado. Esta distruição normal tem desvio padrão maior que o vetor de norma máxima da base do reticulado, fazendo com que a redução pelo paralelepípedo fundamental tenha distribuição indistinguível da uniforme. Além disso, esta construção não revela a geometria da base do reticulado, já que a distribuição normal é esférica. Dada uma mensagem m e uma função de hash H que mapeia textos claros em um elemento pertencente a imagem de f, calcula-se o ponto $y = H(m)$. A assinatura é dada por $\delta = f^{-1}(y)$. Para verificar se a assinatura é válida, basta calcular $f(\delta) = H(m)$. Este tipo de construção

foi proposta por Bellare e Hogaway [Bellare e Rogaway 1995], usando funções permutação com alçapão (*trapdoor permutations*) e modelando H como um orácula aleatório. Assim, é obtido um esquema para assinatura digital com inforjabilidade existencial sobre ataques adaptativos de texto claro escolhido. Para contruir f, a distribuição normal é usada para gerar um ruído e, de maneira que $f(e) = y$, onde $y = v + e$, para um ponto v escolhido uniformemente no reticulado. Assim, a construção é amparada por uma redução de segurança com base no pior caso de problemas em reticulados.

Em linhas gerais, a criptografia baseada em reticulados é construída usando duas funções: $f_A(x) = Ax \pmod q$ - esquema de Ajtai - e $g_A(s, e) = A^T s + e$ - problema LWE - onde a primeira é uma função sobrejetiva e a segunda é uma função injetiva. Em 2012, Micciancio e Peikert [Micciancio e Peikert 2012] mostraram uma forma mais simples, segura e eficiente de inverter g_A e amostrar na pré-imagem de f_A, permitindo a construção de um esquema de assinatura digital mais eficiente. Nesta proposta, o processo de composição das distribuições normais passou a permitir paralelismo (no trabalho de Gentry, Peikert e Vaikuntanathan [Gentry et al. 2008], e trabalhos subsequentes [Stehlé e Steinfeld 2011], esta era uma operação inerentemente sequencial), levando a um ganho concreto considerável. As melhorias descritas neste trabalho podem ser aproveitadas em todas as aplicações que têm como base a inversão da função g_A ou a amostragem na pré-imagem de f_A, portanto, não apenas é importante para assinaturas digitais, como também para construção de encriptação segura no modelo de ataque adaptativo de texto encriptado escolhido (*CCA-secure*).

2.5.5. Outras aplicações

A criptografia baseada em reticulados não apenas é interessante porque resiste a ataques quânticos, mas também porque tem se mostrado uma alternativa flexível para a construção de criptossistemas. Em especial, o problema LWE sobre aneis polinomiais tem se tornado cada vez mais importante, tendo em vista a construção de funções alçapão mais fortes, permitindo uma escolha de parâmetros melhor tanto para segurança quanto para a performance [Micciancio e Peikert 2012].

Gentry [Gentry 2013] faz uma análise sobre quão flexível a criptografia pode ser, levando em consideração não somente a construção de encriptação completamente homomórfica, que permite a computação sobre dados encriptados, como também a questão do controle de acesso. Assim, a criptografia baseada em reticulados parece ser, de acordo com Gentry, uma alternativa viável para explorar os limites da criptomania. Dentre outras aplicações, é possível destacar o seguinte:

- **mapas multilineares**. Emparelhamentos bilineares podem ser utilizados em diferentes contextos, como por exemplo em criptografia baseada em identidades. A generalização do conceito, chamada de mapas multilineares, é bastante útil e, apesar de não haver nenhum esquema proposto por um período, diversas aplicações foram vislumbradas. Utilizando a ideia de ruído, também usada na encriptação homomórfica, Garg, Gentry e Halevi concretizaram a construção de mapas multilineares [Garg et al. 2013a];

- **criptografia baseada em identidades**. Por algum tempo, a única forma de cons-

truir criptografia baseada em identidades era com o uso de emparelhamentos bilineares. Utilizando reticulados, diversas propostas foram feitas [Boneh et al. 2007, Gentry et al. 2008], usando para isso um esquema dual, $\mathcal{E} = \{\text{DualKeyGen}, \text{DualEnc}, \text{DualDec}\}$, em relação ao esquema descrito na seção 2.5.3.3. Especificamente, DualKeyGen calcula a chave privada como sendo o erro e, escolhido pela distribuição normal, enquanto a chave pública é dada por $u = f_A(e)$. Para encriptar um bit b, o algoritmo DualEnc escolhe aleatoriamente s, escolhe x e e' de acordo com a distribuição normal e calcula $c_1 = g_A(s, e)$ e $c_2 = u^T s + e' + b.\lfloor q/2 \rfloor$. O texto encriptado é $\langle c_1, c_2 \rangle$. Por fim, DualDec calcula $b = c_2 - e^T c_1$. Com isso, dada uma função de hash H, modelada como um oráculo aleatório, mapeando identidades em chaves públicas do criptossistema dual, o esquema de encriptação baseada em identidades é construído da seguinte maneira:

  - Configurar. Escolhe-se a chave pública do sistema $A \in \mathbb{Z}_q^{n \times m}$ e a chave mestra como sendo o alçapão s, de acordo com a descrição da seção 2.5.4;
  - Extrair. Dada uma identidade id, calcula-se $u = H(\text{id})$ e a chave de decriptação $e = f^{-1}(u)$, usando o algoritmo de amostragem de pré-imagem com o alçapão s;
  - Encriptar. Dado um bit b, retorne $\langle c_1, c_2 \rangle = \text{DualEnc}(u, b)$;
  - Decriptar. Retorne $\text{DualDec}(e, \langle c_1, c_2 \rangle)$.

- **encriptação funcional**. A encriptação funcional é uma nova forma de encarar a criptografia, abrindo caminho para novas funcionalidades [Lewko et al. 2010]. Neste sistema, uma função pública $f(x, y)$ determina o que um usuário com o conhecimento da chave y pode inferir sobre um texto encriptado, denotado por c_x, de acordo com o parâmetro x. Neste modelo, quem encripta uma mensagem m pode, previamente, escolher que tipo de informação é obtida após a decriptação. Além disso, uma entidade de confiança é responsável por gerar uma chave s_y, que pode ser utilizada para decriptar c_x, obtendo como retorno $f(x, y)$, sem necessariamente revelar informação a respeito de m. Com esta abordagem, é possível definir a criptografia baseada em identidades como um caso especial da encriptação funcional, onde $x = (m, \text{id})$ e $f(x, y) = m$ se e e somente se $y = \text{id}$. Em um trabalho recente [Garg et al. 2013b], foi proposto um esquema de encriptação funcional com base em reticulados, capaz de lidar com qualquer circuito booleano de tamanho polinomial;

- **encriptação baseada em atributos**. Este é um caso especial da encriptação funcional, onde $x = (m, \phi)$ e $f(x, y) = m$ se e somente se $\phi(y) = 1$. Isto é, a decriptação funciona desde que y, os atributos de quem decripta a mensagem, satisfaça o predicado ϕ, de modo que aquele que a encripta pode determinar uma política de acesso (predicado ϕ) para o criptossistema. Existem propostas para realizar este tipo de operação com base no problema LWE [Sahai e Waters 2012] e a construção de mapas multilineares, citada acima, foi utilizada por Sahai e Waters [Gentry et al. 2013] para propor um esquema de encriptação baseada em atributos para qualquer circuito booleano, mostrando mais uma vez a versatilidade da criptografia baseada em reticulados;

- **ofuscação**. Existem resultados negativos que mostram que a ofuscação de código é impossível em um determinado modelo de segurança. Porém, pode-se realizar a ofuscação da melhor maneira possível, levando ao conceito de *indistinguibilidade de ofuscação* (*indistinguishability obfuscation*). O problema LWE foi utilizado para construir este tipo de primitiva [Garg et al. 2013b], que é uma parte da construção de encriptação funcional. Tais construções, portanto, apesar de versáteis, são resultados com uma importância maior pela contribuição teórica do que pela praticidade das aplicações.

Referências

[Ajtai 1996] Ajtai, M. (1996). Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 99–108, New York, NY, USA. ACM.

[Alabbadi e Wicker 1994] Alabbadi, M. e Wicker, S. B. (1994). A digital signature scheme based on linear error-correcting block codes. In *Proc. 4th International Advances in Cryptology Conference – ASIACRYPT '94*, pages 238–348.

[Babai 1986] Babai, L. (1986). On lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, (6).

[Baldi e Chiaraluce 2007] Baldi, M. e Chiaraluce, F. (2007). Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC code. In *IEEE International Symposium on Information Theory – ISIT 2007*, pages 2591–2595, Nice, France. IEEE.

[Baldi et al. 2008] Baldi, M., Chiaraluce, F., e Bodrato, M. (2008). A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Security and Cryptography for Networks – SCN 2008*, volume 5229 of *Lecture Notes in Computer Science*, pages 246–262, Amalfi, Italia. Springer.

[Barbulescu et al. 2013] Barbulescu, R., Gaudry, P., Joux, A., e Thomé, E. (2013). A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. HAL-INRIA technical report, `http://hal.inria.fr/hal-00835446/`.

[Bellare e Goldwasser 1994] Bellare, M. e Goldwasser, S. (1994). The complexity of decision versus search. *SIAM Journal on Computing*, 23:97–119.

[Bellare e Rogaway 1995] Bellare, M. e Rogaway, P. (1995). Random oracles are practical: A paradigm for designing efficient protocols.

[Berger et al. 2009] Berger, T. P., Cayrel, P.-L., Gaborit, P., e Otmani, A. (2009). Reducing key length of the McEliece cryptosystem. In *Progress in Cryptology – Africacrypt 2009*, Lecture Notes in Computer Science, pages 77–97, Gammarth, Tunisia. Springer.

[Berlekamp et al. 1978] Berlekamp, E., McEliece, R., e van Tilborg, H. (1978). On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386.

[Bernstein et al. 2011] Bernstein, D., Lange, T., e Peters, C. (2011). Smaller decoding exponents: ball-collision decoding. In *Advances in Cryptology – Crypto 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760, Santa Barbara, USA. Springer.

[Bernstein 2011] Bernstein, D. J. (2011). List decoding for binary Goppa codes. In *Coding and cryptology—third international workshop, IWCC 2011*, Lecture Notes in Computer Science, pages 62–80, Qingdao, China. Springer.

[Bernstein et al. 2008a] Bernstein, D. J., Buchmann, J., e Dahmen, E. (2008a). *Post-Quantum Cryptography*. Springer, Heidelberg, Deutschland.

[Bernstein et al. 2008b] Bernstein, D. J., Lange, T., e Peters, C. (2008b). Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography Workshop – PQCrypto 2008*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer. http://www.springerlink.com/content/68v69185x478p53g.

[Bernstein et al. 2010] Bernstein, D. J., Lange, T., e Peters, C. (2010). Wild McEliece. In *Selected Areas in Cryptography – SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 143–158, Waterloo, Canada. Springer.

[Bertoni et al. 2007] Bertoni, G., Daemen, J., Peeters, M., e Assche, G. V. (2007). Sponge functions. ECRYPT Hash Workshop 2007. Also available as public comment to NIST from http://www.csrc.nist.gov/pki/HashWorkshop/Public_Comments/2007_May.html.

[Boneh et al. 2007] Boneh, D., Gentry, C., e Hamburg, M. (2007). Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657.

[Braeken et al. 2005] Braeken, A., Wolf, C., e Preneel, B. (2005). A study of the security of unbalanced oil and vinegar signature schemes. In *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 29–43. Springer.

[Brakerski e Vaikuntanathan 2011] Brakerski, Z. e Vaikuntanathan, V. (2011). Efficient fully homomorphic encryption from (standard) lwe. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:109.

[Buchmann et al. 2006] Buchmann, J., Coronado, C., Dahmen, E., Döring, M., e Klintsevich, E. (2006). CMSS– an improved merkle signature scheme. In *Progress in Cryptology – INDOCRYPT 2006, LNCS 4329*, pages 349–363. Springe-Verlag.

[Buchmann et al. 2011a] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., e Rückert, M. (2011a). On the security of the winternitz one-time signature scheme. In *German Research*, pages 1–17.

[Buchmann et al. 2011b] Buchmann, J., Dahmen, E., e Hülsing, A. (2011b). XMSS-a practical secure signature scheme based on minimal security assumptions. In *Cryptology ePrint Archive - Report 2011/484*. ePrint.

[Buchmann et al. 2007] Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., e Vuillaume, C. (2007). Merkle signatures with virtually unlimited signature capacity. In *Applied Cryptography and Network Security - ACNS 2007, LNCS 4521*, pages 31–45. Springer.

[Buchmann et al. 2008] Buchmann, J., Dahmen, E., e Schneider, M. (2008). Merkle tree traversal revisited. In *Proceedings of the 2nd International Workshop on Post-Quantum Cryptography*, pages 63–78. Springer-Verlag.

[Contini et al. 2005] Contini, S., Lenstra, A. K., e Steinfeld, R. (2005). VSH, an Efficient and Provable Collision Resistant Hash Function. Cryptology ePrint Archive, Report 2005/193. `http://eprint.iacr.org/`.

[Courtois et al. 2001] Courtois, N., Finiasz, M., e Sendrier, N. (2001). How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174, Gold Coast, Australia. Springer.

[Courtois et al. 2002] Courtois, N., Goubin, L., Meier, W., daniel Tacier, J., e Lab, C. C. (2002). Solving underdefined systems of multivariate quadratic equations. In *Proceedings of Public Key Cryptography 2002, LNCS 2274*, pages 211–227. Springer-Verlag.

[Ding e Schmidt 2005] Ding, J. e Schmidt, D. (2005). Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security – ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer.

[Dods et al. 2005a] Dods, C., Smart, N., e Stam, M. (2005a). Hash based digital signature schemes. In *Cryptography and Coding*, pages 96–115. Springer Verlag LNCS 3796.

[Dods et al. 2005b] Dods, C., Smart, N., e Stam, M. (2005b). Hash-based digital signature schemes. In *In Cryptography and Coding, LNCS 3796*, pages 96–115. Springer.

[Faugère et al. 2010] Faugère, J.-C., Otmani, A., Perret, L., e Tillich, J.-P. (2010). Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology – Eurocrypt 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298, Nice, France. Springer.

[Gaborit 2005] Gaborit, P. (2005). Shorter keys for code based cryptography. In *International Workshop on Coding and Cryptography – WCC 2005*, pages 81–91, Bergen, Norway. ACM Press.

[Gallager 1963] Gallager, R. G. (1963). Low-density parity-check codes.

[Garey e Johnson 1979] Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.

[Garg et al. 2013a] Garg, S., Gentry, C., e Halevi, S. (2013a). Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17.

[Garg et al. 2013b] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., e Waters, B. (2013b). Candidate indistinguishability obfuscation and functional encryption for all circuits. *IACR Cryptology ePrint Archive*, 2013:451.

[Gentry 2009] Gentry, C. (2009). *A fully homomorphic encryption scheme*. PhD thesis, Stanford University. `crypto.stanford.edu/craig`.

[Gentry 2013] Gentry, C. (2013). Encrypted messages from the heights of cryptomania. In *TCC*, pages 120–121.

[Gentry et al. 2008] Gentry, C., Peikert, C., e Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 197–206, New York, NY, USA. ACM.

[Gentry et al. 2013] Gentry, C., Sahai, A., e Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, pages 75–92.

[Gibson 1996] Gibson, J. K. (1996). The security of the Gabidulin public key cryptosystem. In *Advances in Cryptology – Eurocrypt 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 212–223, Zaragoza, Spain. Springer.

[Goldreich et al. 1997] Goldreich, O., Goldwasser, S., e Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology—CRYPTO '97*, Lecture Notes in Computer Science, pages 112–131. Springer-Verlag.

[Goppa 1970] Goppa, V. D. (1970). A new class of linear error correcting codes. *Problemy Peredachi Informatsii*, 6:24–30.

[Hoffstein et al. 1998] Hoffstein, J., Pipher, J., e Silverman, J. H. (1998). Ntru: A ring-based public key cryptosystem. In *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag.

[Huffman e Pless 2003] Huffman, W. e Pless, V. (2003). *Fundamentals of Error-Correcting Codes*. Cambridge University Press.

[J. Buchmann e Szydlo 2008] J. Buchmann, E. D. e Szydlo, M. (2008). Hash-based digital signature schemes. In *Post-Quantum Cryptography*, pages 35–92. Springer.

[Kipnis et al. 1999] Kipnis, A., Patarin, J., e Goubin, L. (1999). Unbalanced oil and vinegar signature schemes. In Stern, J., editor, *In Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer.

[Kipnis et al. 2003] Kipnis, A., Patarin, J., e Goubin, L. (2003). Unbalanced oil and vinegar signature schemes – extended version.

[Kipnis e Shamir 1998] Kipnis, A. e Shamir, A. (1998). Cryptanalysis of the oil and vinegar signature scheme. In Krawczyk, H., editor, *Advances in Cryptology – Crypto 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Springer.

[Lamport 1979] Lamport, L. (1979). Constructing digital signatures from a one way function. In *SRI International*. CSL-98.

[Lenstra et al. 1982] Lenstra, A. K., Lenstra, H. W., e Lovász, L. (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534.

[Lewko et al. 2010] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., e Waters, B. (2010). Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Gilbert, H., editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer Berlin Heidelberg.

[Lyubashevsky et al. 2010] Lyubashevsky, V., Peikert, C., e Regev, O. (2010). On ideal lattices and learning with errors over rings. *Advances in Cryptology EUROCRYPT 2010*, 6110/2010(015848):1?23.

[MacWilliams e Sloane 1977] MacWilliams, F. J. e Sloane, N. J. A. (1977). *The theory of error-correcting codes*, volume 16. North-Holland Mathematical Library, Amsterdam, The Netherlands.

[Matyas et al. 1985] Matyas, S., Meyer, C., e Oseas, J. (1985). Generating strong one-way functions with cryptographic algorithm. IBM Techn. Disclosure Bull.

[McEliece 1978] McEliece, R. (1978). A public-key cryptosystem based on algebraic coding theory. The Deep Space Network Progress Report, DSN PR 42–44. `http://ipnpr.jpl.nasa.gov/progressreport2/42-44/44N.PDF`. Acesso em: 18 de outubro de 2013.

[Merkle 1987] Merkle, R. (1987). A digital signature based on a conventional encryption function. In *Proceedings of Crypto '87*, pages 369–378. Springer.

[Merkle 1979] Merkle, R. C. (1979). *Secrecy, Authentication, and Public Key Systems*. Stanford Ph.D. thesis.

[Micciancio e Peikert 2012] Micciancio, D. e Peikert, C. (2012). Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval, D. e Johansson, T., editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer Berlin Heidelberg.

[Miller 1986] Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Advances in Cryptology — Crypto '85*, pages 417–426, New York. Springer-Verlag.

[Misoczki et al. 2012] Misoczki, R., Sendrier, N., Tilllich, J.-P., e Barreto, P. S. L. M. (2012). MDPC-McEliece: New McEliece variants from moderate density parity-check codes. Cryptology ePrint Archive, Report 2012/409. `http://eprint.iacr.org/2012/409`.

[Monico et al. 2000] Monico, C., Rosenthal, J., e Shokrollahi, A. (2000). Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory – ISIT 2000*, page 215, Sorrento, Italy. IEEE.

[Morais e Dahab 2012] Morais, E. M. e Dahab, R. (2012). Encriptação homomórfica. SBSeg.

[Nguyen e Regev 2006] Nguyen, P. e Regev, O. (2006). Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In Vaudenay, S., editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer Berlin Heidelberg.

[Niederreiter 1986] Niederreiter, H. (1986). Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166.

[NIST 2007] NIST (2007). Digital Signature Standard (DSS). FIPS PUB-186-2, http://csrc.nist.gov/publications/fips.

[Oliveira e López 2013] Oliveira, A. K. D. S. e López, J. (2013). Implementação em software do esquema de assinatura digital de merkle e suas variantes. SBSeg.

[Otmani et al. 2010] Otmani, A., Tillich, J.-P., e Dallot, L. (2010). Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, 3(2):129–140.

[Patarin 1996] Patarin, J. (1996). Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In Maurer, U., editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg.

[Patarin 1997] Patarin, J. (1997). The oil and vinegar signature scheme. In *Dagstuhl Workshop on Cryptography*. transparencies.

[Patarin e Goubin 1997] Patarin, J. e Goubin, L. (1997). Trapdoor one-way permutations and multivariate polynomials. In *Proc. of ICICS'97, LNCS 1334*, pages 356–368. Springer.

[Patarin et al. 1998] Patarin, J., Goubin, L., e Courtois, N. (1998). Improved algorithms for isomorphisms of polynomials. In *Advances in Cryptology – EUROCRYPT '98 (Kaisa Nyberg, Ed*, pages 184–200. Springer-Verlag.

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

[Peikert 2009] Peikert, C. (2009). Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 333–342, New York, NY, USA. ACM.

[Petzoldt et al. 2010a] Petzoldt, A., Bulygin, S., e Buchmann, J. (2010a). CyclicRainbow – a multivariate signature scheme with a partially cyclic public key. In Gong, G. e Gupta, K., editors, *Progress in Cryptology – Indocrypt 2010*, volume 6498 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg.

[Petzoldt et al. 2010b] Petzoldt, A., Bulygin, S., e Buchmann, J. (2010b). Cyclicrainbow - a multivariate signature scheme with a partially cyclic public key. In Gong, G. e Gupta, K. C., editors, *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 33–48. Springer.

[Petzoldt et al. 2010c] Petzoldt, A., Bulygin, S., e Buchmann, J. (2010c). Selecting parameters for the Rainbow signature scheme. In Sendrier, N., editor, *Post-Quantum Cryptography Workshop – PQCrypto 2010*, volume 6061 of *Lecture Notes in Computer Science*, pages 218–240. Springer Berlin / Heidelberg. Extended Version: `http://eprint.iacr.org/2010/437`.

[Petzoldt et al. 2011] Petzoldt, A., Bulygin, S., e Buchmann, J. (2011). Linear recurring sequences for the UOV key generation. In *International Conference on Practice and Theory in Public Key Cryptography – PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 335–350. Springer Berlin Heidelberg.

[Preneel 1983] Preneel, B. (1983). *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit Leuven.

[Rabin 1978] Rabin, M. O. (1978). *Foundations of secure computation*, chapter Digitalized signatures. Academic Press.

[Regev 2010] Regev, O. (2010). The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204. IEEE Computer Society.

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

[Sahai e Waters 2012] Sahai, A. e Waters, B. (2012). Attribute-based encryption for circuits from multilinear maps. *CoRR*, abs/1210.5287.

[Sendrier 2011] Sendrier, N. (2011). Decoding one out of many. In Yang, B.-Y., editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer Berlin / Heidelberg. 10.1007/978-3-642-25405-5-4.

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

[Stehlé e Steinfeld 2011] Stehlé, D. e Steinfeld, R. (2011). Making ntru as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 27–47, Berlin, Heidelberg. Springer-Verlag.

[Stern 1989] Stern, J. (1989). A method for finding codewords of small weight. *Coding Theory and Applications*, 388:106–133.

[Stern 1995] Stern, J. (1995). Can one design a signature scheme based on error-correcting codes? *Lecture Notes in Computer Science*, 917:424–??

[Szydlo 2003] Szydlo, M. (2003). Merkle tree traversal in log space and time. In *Prepint version, 2003*.

[Tanner 2001] Tanner, R. M. (2001). Spectral graphs for quasi-cyclic LDPC codes. In *IEEE International Symposium on Information Theory – ISIT 2001*, page 226, Washington, DC, USA. IEEE.

[Thomae 2012] Thomae, E. (2012). A generalization of the Rainbow band separation attack and its applications to multivariate schemes. Cryptology ePrint Archive, Report 2012/223. `http://eprint.iacr.org/2012/223`.

[Umaña e Leander 2010] Umaña, V. G. e Leander, G. (2010). Practical key recovery attacks on two McEliece variants. In *International Conference on Symbolic Computation and Cryptography – SCC 2010*, Egham, UK. Springer.

[Wieschebrink 2006] Wieschebrink, C. (2006). Two NP-complete problems in coding theory with an application in code based cryptography. In *IEEE International Symposium on Information Theory – ISIT 2006*, pages 1733–1737, Seattle, USA. IEEE.

[Winternitz 1983] Winternitz, R. S. (1983). Producing a one-way hash function from des. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 203–207. Plenum.

[Wolf e Preneel 2005] Wolf, C. e Preneel, B. (2005). Taxonomy of public key schemes based on the problem of multivariate quadratic equations. *IACR Cryptology ePrint Archive*, 2005:77.

[Yasuda et al. 2012] Yasuda, T., Sakurai, K., e Takagi, T. (2012). Reducing the key size of Rainbow using non-commutative rings. In *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 68–83. Springer.

