UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Gustavo Zambonin

OTIMIZAÇÃO DE DESEMPENHO DO ESQUEMA DE ASSINATURA DIGITAL WINTERNITZ

Florianópolis

2018

Gustavo Zambonin

OTIMIZAÇÃO DE DESEMPENHO DO ESQUEMA DE ASSINATURA DIGITAL WINTERNITZ

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciência da Computação para a obtenção do Grau de Bacharel em Ciência da Computação.
Orientador: Prof. Ricardo Felipe Custódio, Dr.
Coorientador: Prof. Daniel Panario, Dr.

Florianópolis

2018

Gustavo Zambonin

OTIMIZAÇÃO DE DESEMPENHO DO ESQUEMA DE ASSINATURA DIGITAL WINTERNITZ

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de "Bacharel em Ciência da Computação", e aprovado em sua forma final pelo Programa de Graduação em Ciência da Computação.

Florianópolis, 11 de junho 2018.

Prof. Rafael Luiz Cancian, Dr.
Coordenador

Banca Examinadora:

Prof. Ricardo Felipe Custódio, Dr.
Orientador

Prof. Daniel Panario, Dr.
Coorientador

Lucas Pandolfo Perin, Me.
Universidade Federal de Santa Catarina

Às mulheres em minha vida

AGRADECIMENTOS

. . .

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

Neal Stephenson (*Cryptonomicon*, 1999)

RESUMO

Algoritmos utilizados em assinaturas digitais atualmente — como RSA e ECDSA — têm sua segurança baseada na dificuldade de se fatorar números muito grandes ou na obtenção de logaritmos discretos. Este tipo de cômputo pode ser eficientemente realizado por um computador quântico, utilizando algoritmos já conhecidos. Deste modo, para manter o ambiente de assinaturas digitais seguro, é necessário oferecer alternativas pós-quânticas. Este trabalho busca apresentar esquemas baseados apenas em funções de resumo criptográfico, cuja segurança é baseada apenas na resistência à colisões da função escolhida, com o intuito de discutir uma nova otimização para a família de esquemas de assinatura Winternitz. Particularmente, esta proposta introduz um parâmetro de compensação, a fim de reduzir o tempo de geração ou verificação de uma dada assinatura.

Palavras-chave: criptografia pós-quântica, função de resumo criptográfico, assinatura digital, esquema de assinatura digital única Winternitz

ABSTRACT

Algorithms currently used in digital signature schemes, such as RSA and ECDSA, feature security proofs based on the difficulty of calculating large integer factorizations or discrete logarithms. This kind of computation can be achieved through a sufficiently powerful quantum computer running already known algorithms. Hence, to maintain the security *status quo*, it is imperative to offer post-quantum alternatives, that is, schemes resistant to quantum computers. This work explores hash-based digital signature schemes, in which security is reduced to the collision resistance of a chosen hash function, showing that the construction of such secure schemes is independent from hard problems from number theory or algebra. Finally, we introduce an optimization for the Winternitz one-time signature scheme family in the form of a tradeoff parameter, enabling reduced execution time for signature generation or verification.

Keywords: post-quantum cryptography, cryptographic hash function, digital signature, Winternitz one time signature scheme

LISTA DE FIGURAS

Figura 1 Diagrama de Venn das resistências desejáveis para uma função de resumo no contexto de assinaturas digitais. 41
Figura 2 Estrutura da construção esponja, onde $i, j \in \mathbb{N}^*$. 42
Figura 3 Funcionamento típico de um esquema de assinatura digital. . . 44
Figura 4 Etapas de assinatura e verificação do WOTS, para $0 \leq i \leq t-1$. 48
Figura 5 Construção Matyas-Meyer-Oseas. 50
Figura 6 Assinatura no HORS, para $0 \leq i \leq k-1$, $0 \leq j,l \leq t-1$. 52
Figura 7 Árvore de Merkle T, com altura $H = 2$. 53
Figura 8 Caminho de autenticação para a folha de índice $i = 3$. 57
Figura 9 Caminho de autenticação para a folha $j = 3 \cdot 2^{h_0} + 5$, em uma hiper-árvore com $d = 2, h_0 = h_1 = 3$. 61
Figura 10 Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1979 até 1997. . 66
Figura 11 Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1998 até 2018. . 67
Figura 12 Histograma normalizado de $\mu(\mathcal{B}_1^i)$, com 50 classes. 72
Figura 13 Valores de R para que modificações de $\mathcal{B}_1^i$ sejam efetivas. 72
Figura 14 Efeitos da aplicação das otimizações propostas no WOTS. . . . 74

LISTA DE TABELAS

Tabela 1 *Bits* inutilizados em $\mathcal{B}_2$ para vários valores de w e m. 70
Tabela 2 Tempo de execução para otimizações incluídas no XMSS. 75

LISTA DE ABREVIATURAS E SIGLAS

SHA *Secure Hash Algorithm* 33
ECDSA *Elliptic Curve Digital Signature Algorithm* 34
AES *Advanced Encryption Standard* 35
NIST *National Institute of Standards and Technology* 35
DES *Data Encryption Standard* 35
RSA *Rivest—Shamir—Adleman* 37
LD-OTS *Lamport—Diffie One-time Signature Scheme* 45
WOTS *Winternitz One-time Signature Scheme* 45
MSS *Merkle Signature Scheme* 45
WOTS-T *Winternitz One-time Signature Scheme with Tight security* ... 48
IETF *Internet Engineering Task Force* 48
WOTS-LM *Leighton—Micali—Winternitz One-time Signature Scheme* .. 48
HORS *Hash to Obtain Random Subset* 51
HORST *Hash to Obtain Random Subset with Trees* 53
SPR-MSS *Second Preimage Resistant Merkle Signature Scheme* 58
XMSS *eXtended Merkle Signature Scheme* 58
WOTS-PR *Winternitz One-time Signature Scheme with Pseudorandom Functions* 58
CMSS *Coronado—Merkle Signature Scheme* 61
GMSS *Generalized Merkle Signature Scheme* 61
XMSS-MT *eXtended Merkle Signature Scheme with Multi-Trees* 61
BDS *Buchmann—Dahmen—Schneider* 61
XMSS-T *eXtended Merkle Signature Scheme with Tight security* 66
SPHINCS *Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures* 67

LISTA DE SÍMBOLOS

$\lvert\omega\rvert$	Tamanho da palavra ω	34
$\mathbb{F}_n$	Corpo de Galois de ordem n	35
$\overset{\curvearrowright}{\ll}$	Deslocamento circular à esquerda de *bits*	36
$\lvert\rvert$	Concatenação de palavras	43
$\xleftarrow{\$}$	Seleção aleatória	46
$\Sigma^{(l,t)}$	Todas as t-tuplas de palavras de tamanho l construídas com Σ	46
f^{α}	Aplicação de f α vezes	47
$\gg$	Muito maior que	51

LISTA DE ALGORITMOS

Algoritmo 1 Codificação do AES. 36
Algoritmo 2 Decodificação do AES. 37
Algoritmo 3 Funcionamento de TREEHASH$_h$. 54
Algoritmo 4 TREEHASH para o cálculo de $T_{H,0}$. 55
Algoritmo 5 Travessia de árvores de Merkle BDS. 63

SUMÁRIO

1 INTRODUÇÃO 29
1.1 OBJETIVOS 31
1.1.1 Objetivo geral 31
1.1.2 Objetivos específicos 31
1.2 TRABALHOS RELACIONADOS 31
1.3 CONTRIBUIÇÕES DESTE TRABALHO 32
2 PRIMITIVAS CRIPTOGRÁFICAS 33
2.1 CRIPTOGRAFIA SIMÉTRICA E ASSIMÉTRICA 33
2.1.1 A cifra AES 35
2.1.2 O criptossistema RSA 37
2.2 FUNÇÕES DE RESUMO CRIPTOGRÁFICO 39
2.2.1 Construção esponja 41
2.3 ESQUEMAS DE ASSINATURA DIGITAL 43
3 ASSINATURA DIGITAL BASEADA EM FUNÇÕES DE RESUMO CRIPTOGRÁFICO 45
3.1 ESQUEMAS DE ASSINATURA ÚNICA 45
3.1.1 LD-OTS 45
3.1.2 WOTS 47
3.1.3 WOTS+ 49
3.1.4 HORS 51
3.2 ESQUEMAS BASEADOS EM ÁRVORES DE MERKLE 53
3.2.1 MSS 55
3.2.2 XMSS 58
3.2.3 XMSS-MT 60
3.2.3.1 Travessia de árvores de Merkle BDS 62
3.2.3.2 Descrição do esquema 64
3.3 VISÃO TEMPORAL 66
4 OTIMIZAÇÃO DO ESQUEMA WINTERNITZ 69
4.1 WOTS-B 69
4.2 WOTS-R 71
4.3 RESULTADOS 73
4.3.1 Impacto em esquemas baseados em árvores de Merkle 74
5 CONCLUSÃO 77
5.1 TRABALHOS FUTUROS 77
REFERÊNCIAS 79

1 INTRODUÇÃO

A aplicação de protocolos criptográficos é essencial no contexto da validação e proteção de quaisquer comunicações realizadas por um conjunto de entidades, sejam estas dispositivos eletrônicos ou indivíduos, em virtude da possível criticidade e sensibilidade atribuídas aos dados transmitidos. Esquemas de assinatura digital são comumente utilizados para assegurar este processo de maneira formal (GOLDREICH, 2004, Seção 6.1), através da autenticidade e não-repúdio do remetente e certeza da integridade dos dados, a fim de traduzir o resguardo provido por uma assinatura de próprio punho no mundo real.

Na prática, a maior parte destes esquemas utilizam como alicerce algorítmico criptossistemas assimétricos baseados em problemas "difíceis" da teoria dos números, como a fatoração de inteiros ou resolução do logaritmo discreto. Este fato provê a segurança necessária para os esquemas em computadores eletrônicos, por conta da inexistência de algoritmos que resolvem estes problemas em tempo polinomial. Entretanto, em computadores quânticos, algoritmos dessa forma já existem — em especial, o algoritmo de Shor (SHOR, 1997) — efetivamente tornando estes esquemas clássicos inseguros neste novo contexto.

Para combater esta situação, a criptografia pós-quântica encarrega-se de buscar algoritmos criptográficos cuja segurança é considerada "suficiente", mesmo na utilização de um hipotético computador quântico para ataques especializados. Esta área conta com diversas abordagens: a criptografia baseada em reticulados, polinômios de múltiplas variáveis sobre um corpo finito, teoria de códigos, morfismos entre curvas elípticas supersingulares e criptossistemas simétricos. Entretanto, alguns destes métodos não podem ser utilizados no contexto de esquemas de assinatura digital, e para outros, a inexistência de reduções de segurança formais e o tamanho das chaves impossibilitam a utilização destes em aplicações práticas (BERNSTEIN; LANGE, 2017).

Não obstante, uma abordagem adicional de esquema de assinatura digital resistente a computadores quânticos é baseada apenas em funções de resumo criptográfico, construídas a partir de funções de mão única (KATZ; KOO, 2005). De fato, estas funções, desde que apresentem requisitos de segurança como resistência à segunda pré-imagem e/ou à colisões, são necessárias e suficientes para a construção de esquemas bem comportados e seguros (ROMPEL, 1990). Visto que estas funções são estudadas exaustivamente por conta de sua vasta presença em diversos âmbitos da segurança da informação, reduções de segurança são mais comuns em relação a outras abordagens pós-quânticas, e tamanhos de chaves e assinaturas são menos proibitivos.

Esquemas de assinatura digital baseados em funções de resumo criptográfico consistem da utilização de um esquema de assinatura digital única, onde apenas uma mensagem pode ser assinada de modo seguro, possivelmente combinados à estrutura de dados chamada de árvore de Merkle (MERKLE, 1989), que abriga pares de chaves de diversas instâncias do esquema supracitado como suas folhas, e reduz a verificação destes para uma única chave, codificada em sua raiz. Esta árvore é construída com a concatenação de resumos criptográficos do conteúdo dos nós, habilitando assim a assinatura de diversas mensagens. Como uma função específica não é necessária, é possível obter uma grande variedade de esquemas, garantindo a versatilidade destas abordagens.

Embora os esquemas iniciais tenham sido construídos sem atenção particular à eficiência de modo geral (*e.g.* o esquema de assinatura única de Lamport (LAMPORT, 1979) assina apenas um *bit* de informação em sua forma mais simples), muitos resultados práticos demonstram a redução contínua do tempo de verificação da assinatura, tamanho e tempo para geração do par de chaves e assinatura, bem como avanços teóricos que possibilitam a utilização de funções com requisitos de segurança mínimos (HüLSING, 2013), garantem o conceito de sigilo encaminhado (BUCHMANN; DAHMEN; HüLSING, 2011) e a ausência do gerenciamento de estado (BERNSTEIN et al., 2015).

Neste trabalho, é enfatizado o estudo do esquema de assinatura digital única Winternitz, no qual o resumo criptográfico da mensagem a ser assinada define diretamente a complexidade dos passos de geração e verificação da assinatura. De acordo com este comportamento, é apresentada uma adaptação para o esquema na forma de um parâmetro adicional, que habilita a redução do cômputo de verificação de assinatura em troca do aumento deste na geração da assinatura, ou vice-versa.

Ambientes nos quais existem limitações de processamento para uma destas etapas, como redes de sensores sem fio, podem configurar instâncias do esquema Winternitz a fim de otimizar sua comunicação segura, aumentando o tempo de execução do cômputo para a geração de assinaturas, geralmente feita em dispositivos com poder de processamento amplificado. Ademais, as consequências desta otimização são verificadas em esquemas mais complexos, baseados em árvores de Merkle, que também podem ser aplicados a meios com restrições similares.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Apresentar um estudo detalhado sobre os principais esquemas de assinatura digital baseados em funções de resumo criptográfico. Em especial, um estudo detalhado sobre o esquema de assinatura digital única Winternitz é realizado, contextualizando-o junto aos algoritmos previamente descritos. Este processo tem o intuito de fundamentar a otimização proposta, que afeta o tempo de execução da criação ou verificação de uma assinatura.

1.1.2 Objetivos específicos

i. Descrever os esquemas de assinatura digital única LD-OTS e WOTS, sua variante WOTS+, e o esquema de poucas assinaturas HORS;

ii. Descrever os esquemas de assinatura digital baseado em árvores de Merkle MSS, XMSS e XMSS-MT;

iii. Apresentar a otimização para a família WOTS e discutir consequências desta modificação no contexto de esquemas baseados em árvores de Merkle.

1.2 TRABALHOS RELACIONADOS

Artigos na literatura recente consistentes com o tema deste trabalho focam na redução do tempo de execução da verificação da assinatura, através da definição de novos esquemas variantes. Em especial, o esquema proposto em (CRUZ; YATANI; KAJI, 2016) torna a complexidade deste passo previsível através de uma função que mapeia a mensagem para uma tupla com propriedades especiais; em (STEINWANDT; VILLáNYI, 2008), a mensagem é modificada a fim de gerar um resumo criptográfico cuja codificação *run-length* é otimizada; e em (MCGREW; CURCIO; FLUHRER, 2018), um parâmetro é adicionado que marca o número de *bits* inutilizados na assinatura, a fim de que estes não sejam verificados de maneira supérflua.

Neste trabalho, a estrutura do esquema base não é modificada, e um pré-processamento da mensagem é feito alternativamente, permitindo a escalabilidade de instâncias de esquemas Winternitz com quaisquer parâmetros, introduzindo apenas um pequeno custo na etapa de assinatura da mensagem.

1.3 CONTRIBUIÇÕES DESTE TRABALHO

Uma versão resumida deste trabalho (PERIN et al., 2018) será publicada como um artigo completo em *22th IEEE Symposium on Computers and Communications* (ISCC 2018), sob o nome *Tuning the Winternitz Hash-Based Digital Signature Scheme*. Ainda, o artigo pode ser consultado no Apêndice **??**. Este trabalho busca adicionalmente fornecer explicações didáticas para vários esquemas baseados em funções de resumo criptográfico presentes na literatura, bem como uma abordagem complementar aos argumentos providos pelo artigo, no Capítulo 4.

2 PRIMITIVAS CRIPTOGRÁFICAS

Neste capítulo, são explicados os conceitos necessários a fim de entender inteiramente um esquema de assinatura digital, bem como outros algoritmos discutidos ao longo do trabalho. A organização das seções é semelhante à seguida em (VON ZUR GATHEN, 2015, Capítulo 2). Os conceitos de criptografia simétrica e assimétrica são apresentados na Seção 2.1, e uma simples comparação entre os mesmos é realizada, bem como exemplos apresentados na forma dos algoritmos AES e RSA, amplamente padronizados e implementados em diversas formas de comunicação segura, nas Subseções 2.1.1 e 2.1.2. Funções de resumo criptográfico, utilizadas para que o processo de assinatura seja menos custoso e mais seguro, são discutidas na Seção 2.2, e um exemplo de construção teórica por trás deste tipo de função na Subseção 2.2.1; por fim, a definição formal de um esquema de assinatura digital é apresentada na Seção 2.3, agregando estas noções.

2.1 CRIPTOGRAFIA SIMÉTRICA E ASSIMÉTRICA

Define-se criptografia como a criação e análise de protocolos matemáticos que habilitam comunicação segura, através de um canal inseguro, entre duas ou mais entidades. Implementações destes, comumente chamadas de algoritmos criptográficos, podem ser parametrizadas por uma chave, que habilita a transformação do texto plano para texto cifrado de acordo com esta, de maneira individual e inteligível, mas a fim de tornar o resultado irrecuperável sem a apresentação da chave correspondente. De acordo com esta característica, estes algoritmos são classificados em duas grandes famílias.

Sistemas que utilizam a mesma chave para as operações de codificação e decodificação são chamados de simétricos. Nesta situação, a chave representa um segredo compartilhado entre entidades desejando estabelecer comunicação segura. Como o ato de compartilhar este segredo necessita, por si próprio, de um canal seguro, este aspecto é uma desvantagem destes criptossistemas.

Cifras de bloco ou de fluxo são considerados exemplos convencionais de criptossistemas simétricos. A construção destes geralmente é feita através do encadeamento de operações binárias e matemáticas favoráveis para computadores, assim permitindo aos algoritmos um desempenho altíssimo. Utilizando estas como alicerce, é possível construir funções de resumo criptográfico: por exemplo, a construção Merkle—Damgård, base para as funções MD5, SHA1 e SHA2, utiliza uma função de compressão de mão única, obtida

a partir de uma cifra de bloco (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Algoritmo 9.41)[1].

No contexto da computação quântica, estes sistemas são ameaçados pelo algoritmo de Grover (GROVER, 1996), que possibilita a busca de elementos em conjuntos em tempo reduzido. Por outra forma, suponha que é desejável inverter uma função $f : A \longrightarrow B$, a partir de um elemento $b \in B$. Classicamente, como não existe informação qualquer sobre a função, é necessário calcular f para cada um dos elementos de seu domínio, *i.e.* $|A|$ vezes. Entretanto, Grover permite que esta busca seja feita em $|A|^{\frac{1}{2}}$ operações. Assim, um algoritmo criptográfico simétrico que toma o lugar de f, neste exemplo, precisa de parâmetros de segurança maiores, a fim de igualar a dificuldade da busca em se tratando de um computador clássico.

Em contrapartida, a criptografia assimétrica, ou criptografia de chaves públicas, engloba os algoritmos que utilizam um par de chaves: a chave privada, conhecida apenas pela entidade que a gerou, e a chave pública, distribuída livremente. Podem ser representadas por, respectivamente, $\mathcal{S}_k$ e $\mathcal{P}_k$. Isto possibilita o uso livre de $\mathcal{P}_k$ para a comunicação segura com o detentor da chave sem a necessidade de um canal seguro, em virtude da construção dos algoritmos.

A ideia foi introduzida abstratamente em (DIFFIE; HELLMAN, 1976) e tem como exemplos algoritmos como RSA, ElGamal e ECDSA. Diferentemente dos algoritmos simétricos, estes sistemas utilizam operações mais robustas e, portanto, de desempenho reduzido. Assim, a utilização convencional destes dois tipos de criptografia em protocolos ocorre através da codificação de uma chave simétrica, responsável por cifrar um documento de tamanho não trivial, com uma chave pública, e transmissão desta "chave cifrada" sem a necessidade de um canal seguro.

A segurança de sistemas assimétricos depende da "dificuldade" computacional de determinar uma chave privada a partir da chave pública, e também do armazenamento de $\mathcal{S}_k$ em um lugar seguro. Problemas em teoria de números e álgebra que atualmente não admitem soluções em tempo polinomial são comumente utilizados como base para algoritmos assimétricos. Porém, com a introdução de um computador quântico, estes problemas podem ser resolvidos de maneira significativamente mais rápida, como visto em (SHOR, 1997).

[1] Este processo é explorado na Subseção 3.1.3.

2.1.1 A cifra AES

Originalmente publicado como Rijndael, o algoritmo conhecido como AES é resultado de um esforço de padronização para um sistema criptográfico seguro, finalizado ao término do século XX pelo NIST (DWORKIN et al., 2001), a fim de substituir a cifra DES. Definido como uma cifra de blocos iterativa, opera sobre uma matriz de estado A, onde $A_{i,j} \in \mathbb{F}_{2^8}$[2], $0 \leq i,j \leq 3$, a partir de uma chave K de tamanho $n \in \{128, 192, 256\}$. Consiste em aplicações sequenciais de quatro operações ordenadas sobre A. A quantidade destas aplicações, denominadas rodadas (n_r), depende diretamente do tamanho da chave: $n = 128 \rightarrow n_r = 10, n = 192 \rightarrow n_r = 12, n = 256 \rightarrow n_r = 14$.

Uma rotina de expansão de chave existe para que K seja propagada em todas as rodadas com valores derivados, porém variados entre si. A presença desta rotina é fundamentada pela construção abstrata no qual o Rijndael é baseado, chamada de rede de substituição-permutação, onde o estado inicial é primeiramente modificado com uma chave de rodada. As operações e rotinas serão descritas abaixo.

i. SUBBYTES: realiza-se a reposição de $A_{i,j}$ pelo seu valor correspondente em uma caixa de substituição, também chamada de *S-box*, construída a partir de uma transformação afim em $A_{i,j}^{-1}$. A escolha da linha e coluna nesta matriz é feita, respectivamente, a partir dos *nibbles* mais e menos significativos do elemento.

ii. SHIFTROWS: cada linha de A, A_i, é deslocada circularmente à esquerda i vezes.

iii. MIXCOLUMNS: cada coluna de A, A_j, é multiplicada pelo polinômio $c = 3x^3 + x^2 + x + 2$, módulo $x^4 + 1$, para que o resultado ainda seja um polinômio de grau máximo 3, apto a ser representado na coluna.

iv. ADDROUNDKEY: a operação de ou exclusivo *bit* a *bit* é aplicada entre A e o bloco da chave referente à rodada.

v. KEYEXPANSION consiste da criação de um conjunto de palavras K_e de 32 *bits*. Tome $\ell = \frac{n}{32}$, e assumindo que é necessário criar palavras suficientes para utilização em todas as rodadas do algoritmo, então $t = \ell \cdot (n_r + 1)$ e $K_e = \{k_0, \ldots, k_{t-1}\}$. A lista de constantes RC com elementos em $\mathbb{F}_{2^8}$ é definida pela recursão

$$RC_0 = x^0, \quad RC_1 = x^1, \quad RC_j = x \cdot RC_{j-1},\ j \geq 2. \tag{2.1}$$

[2]Definido pelo polinômio irredutível $m(x) = x^8 + x^4 + x^3 + x + 1$. Adições e multiplicações sobre elementos em corpos da forma $\mathbb{F}_{2^n}$ são análogas a operações computacionais.

Inicialmente, K é dividida em ℓ palavras, que compõem o começo de K_e. Para os elementos restantes, ou seja, $\forall i \geq \ell$,

$$k_i = k_{i-\ell} + \begin{cases} \text{SUBBYTES}(k_{i-1} \overset{\curvearrowright}{\ll} 8) + RC_{\frac{i}{\ell}}, \text{ se } i \equiv 0 \pmod 4, \\ \text{SUBBYTES}(k_{i-1}), \text{ se } \ell = 8 \text{ e } i \equiv 4 \pmod 8, \\ k_{i-1}, \text{ caso contrário.} \end{cases} \tag{2.2}$$

Assim, uma função que criptografa uma mensagem m e retorna um texto cifrado c pode ser representada pelo Algoritmo 1. A mensagem é primeiramente codificada em A da seguinte maneira: $A_{i,j} = m_{i+4j}, 0 \leq i, j \leq 3$. Note que MIXCOLUMNS é ignorado na última rodada, a fim de facilitar a reversibilidade da cifra.

Algoritmo 1 Codificação do AES.

Entrada: $m \in \{0,1\}^{128}$, $K \in \cup_{n \in \{128,192,256\}} \{0,1\}^n$ ▷ texto plano e chave
Saída: $c \in \{0,1\}^{128}$ ▷ texto cifrado

```
A ← m
{k_0 ... k_{(n_r+1)·ℓ}} ← KEYEXPANSION(K)
A ← ADDROUNDKEY(A, {k_0, ..., k_{ℓ-1}})
para i ← 1 até n_r − 1 faça
    A ← SUBBYTES(A)
    A ← SHIFTROWS(A)
    A ← MIXCOLUMNS(A)
    A ← ADDROUNDKEY(A, {k_{i·ℓ}, ..., k_{(i+1)·ℓ-1}})
fim para
A ← SUBBYTES(A)
A ← SHIFTROWS(A)
A ← ADDROUNDKEY(A, {k_{n_r·ℓ}, ..., k_{(n_r+1)·ℓ-1}})
c ← A
```

Para que a cifra seja caracterizada como simétrica, é preciso criar uma função que faça o inverso do procedimento acima. Assim, suas etapas precisam ser modificadas de acordo.

i. INVSHIFTROWS: cada linha de A, A_i, é deslocada circularmente à direita i vezes.

ii. INVSUBBYTES: é necessário computar a transformação afim inversa para cada elemento $A_{i,j}$, e depois calcular sua inversa multiplicativa.

iii. INVMIXCOLUMNS: cada coluna de A, A_j, é multiplicada pela inversa multiplicativa $d = c^{-1}$, obtida por

$$\begin{aligned}(3x^3 + x^2 + x + 2) \times d &\equiv 1 \pmod{x^4 + 1} \\ d &= 11x^3 + 13x^2 + 9x + 14.\end{aligned} \tag{2.3}$$

Por fim, o resultado é representado pelo Algoritmo 2. Note a mudança da ordem das etapas, e a utilização invertida de K_e. Finalmente, c é codificado no estado de maneira análoga ao algoritmo anterior.

Algoritmo 2 Decodificação do AES.

Entrada: $c \in \{0,1\}^{128}$, $K \in \cup_{n \in \{128,192,256\}} \{0,1\}^n$ ▷ texto cifrado e chave
Saída: $m \in \{0,1\}^{128}$ ▷ texto plano

```
A ← c
{k_0 ... k_{(n_r+1)·ℓ}} ← KEYEXPANSION(K)
A ← ADDROUNDKEY(A, {k_{n_r·ℓ}, ..., k_{(n_r+1)·ℓ−1}})
para i ← n_r − 1 até 1 faça
  A ← INVSHIFTROWS(A)
  A ← INVSUBBYTES(A)
  A ← ADDROUNDKEY(A, {k_{i·ℓ}, ..., k_{(i+1)·ℓ−1}})
  A ← INVMIXCOLUMNS(A)
fim para
A ← INVSHIFTROWS(A)
A ← INVSUBBYTES(A)
A ← ADDROUNDKEY(A, {k_0, ..., k_{ℓ−1}})
m ← A
```

Discussões detalhadas sobre as etapas do algoritmo, a fundamentação algébrica por trás deste, escolhas feitas em cada etapa, criptoanálise relacionada e outros tópicos avançados podem ser consultados junto ao livro original de descrição do Rijndael (DAEMEN; RIJMEN, 2002).

2.1.2 O criptossistema RSA

O algoritmo conhecido como RSA (RIVEST; SHAMIR; ADLEMAN, 1978) é uma implementação de criptografia assimétrica amplamente utilizada. É baseado na dificuldade de fatorar o produto de dois números primos suficientemente grandes[3]. Em virtude de seu baixo desempenho computacio-

[3] O algoritmo é baseado no problema RSA, definido como realizar uma operação de chave privada no algoritmo RSA utilizando apenas $\mathcal{P}_k$. Acredita-se que este problema seja equivalente

nal, geralmente apenas um resumo criptográfico da mensagem desejada é codificado por este algoritmo, e sua transmissão é realizada junto à mensagem original, de forma concatenada. Abaixo, uma descrição do funcionamento do algoritmo. Tome $\phi(x)$ como a função totiente de Euler, que representa a quantidade de números relativamente primos a x.

Geração de chaves. Gere dois números primos p,q aleatoriamente, suficientemente grandes e de tamanhos similares. Compute $n = pq$ e $\phi(n) = (p-1)(q-1)$. Selecione um número aleatório e relativamente primo a $\phi(n)$. Então, use o algoritmo de Euclides estendido para computar d tal que $ed \equiv 1 \pmod{\phi(n)}$, *i.e.* a inversa multiplicativa modular de e. Finalmente,

$$\begin{aligned} \mathcal{S}_k &= d, \\ \mathcal{P}_k &= (n,e). \end{aligned} \tag{2.4}$$

Codificação. Para criptografar uma mensagem, é necessário obter $\mathcal{P}_k$ da entidade que age como destinatário. Então, a mensagem m é transformada em um inteiro no intervalo $[0, n-1]$ através de uma função de preenchimento. O texto cifrado

$$c = m^e \pmod{n} \tag{2.5}$$

é calculado através de um algoritmo como a exponenciação quadrática e enviado para a entidade desejada.

Decodificação. O receptor da mensagem calcula

$$m = c^d \pmod{n}. \tag{2.6}$$

Demonstração. Para demonstrar $m^{ed} \equiv m \pmod{n}$, é suficiente mostrar que $m^{ed} \equiv m \pmod{p}$ e $m^{ed} \equiv m \pmod{q}$, pelo Teorema Chinês do Resto. Se $m \equiv 0 \pmod{p}$, então $mdc(m,p) = p$ e certamente $m^{ed} \equiv 0 \equiv m \pmod{n}$. Se $m \not\equiv 0 \pmod{p}$, então $mdc(m,p) = 1$ e pelo Pequeno Teorema de Fermat, $m^{p-1} \equiv 1 \pmod{p}$. Reescrevendo o produto ed como

$$ed = 1 + y\phi(n) = 1 + y(p-1)(q-1),\ y \in \mathbb{N}, \tag{2.7}$$

à fatoração de inteiros (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Fato 3.30).

então

$$m^{ed} \equiv m^{1+y(p-1)(q-1)} \equiv (m^{p-1})^{y(q-1)} m \equiv 1^{y(q-1)} m \equiv m \pmod{p}. \quad (2.8)$$

Analogamente, o processo acima pode ser aplicado para o inteiro q. Portanto, $\forall m \in \mathbb{N}$, $m^{ed} \equiv m \pmod{n}$.

Observe que a descrição acima, embora funcional, apresenta sérios problemas de segurança se aplicada de maneira ingênua. Defina a noção de segurança semântica como, dado um texto cifrado, a impossibilidade de revelar informações quaisquer sobre seu texto plano correspondente (GOLDWASSER; MICALI, 1982). Observe, por exemplo, que não existe qualquer fator aleatório na codificação da mensagem, caracterizando-a como determinística e habilitando uma entidade maliciosa a aplicar um ataque de texto plano escolhido, *i.e.* a codificação de múltiplas mensagens a fim de descobrir informações sobre o algoritmo baseado em semelhanças entre as mensagens cifradas, portanto semanticamente inseguro. Este e outros ataques (**??**) são mitigados com a aplicação de uma função de preenchimento à mensagem (**??**).

2.2 FUNÇÕES DE RESUMO CRIPTOGRÁFICO

Funções criadas com o intuito de resumir dados, ou seja, reduzir uma mensagem de tamanho arbitrário para uma palavra pequena e identificável, podem possuir várias propriedades, apresentadas abaixo de acordo com (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Seção 9.2). Tome uma função $\mathcal{H} : X \longrightarrow Y$. Comumente, os elementos de Y são chamados de resumos. É importante notar que um problema é considerado "difícil", ou computacionalmente inviável, quando o tempo ou recursos gastos para esta computação excedem a validade ou utilidade da informação desejada.

(i) A função deve ser necessariamente *determinística*;

(ii) O cálculo de todo resumo deve ser *computacionalmente fácil*[4];

(iii) $\mathcal{H}$ pode apresentar *compressão*, ou seja, o mapeamento de uma palavra $x \in X$ de tamanho arbitrário para um resumo $y \in Y$ de tamanho fixo;

(iv) $\mathcal{H}$ pode apresentar *resistência à pré-imagem* (PRE), caracterizada pelo seguinte comportamento: fornecido um resumo $h \in Y$, é computacio-

[4]Note que existem funções de desempenho reduzível, como `bcrypt`, prevenindo ataques de força bruta a serviços que armazenam resumos de dados sensíveis, *e.g.* senhas.

nalmente inviável achar alguma mensagem original $m \in X$ que gere h através de $\mathcal{H}(m) = h$;

(v) $\mathcal{H}$ pode apresentar *resistência à segunda pré-imagem* (SEC), caracterizada pelo seguinte comportamento: fornecida uma mensagem $m_0 \in X$, é computacionalmente inviável achar uma mensagem $m_1 \in X$ tal que $m_0 \neq m_1$ e $\mathcal{H}(m_0) = \mathcal{H}(m_1)$;

(vi) $\mathcal{H}$ pode apresentar *resistência à colisões* (COL), caracterizada pelo seguinte comportamento: é computacionalmente inviável encontrar duas mensagens m_0, $m_1 \in X$ e $m_0 \neq m_1$, de tal forma que $\mathcal{H}(m_0) = \mathcal{H}(m_1)$;

(vii) $\mathcal{H}$ pode ser parametrizada por uma chave k. Este comportamento é representado por $\mathcal{H}_k$.

A classificação de $\mathcal{H}$ é realizada de acordo com a presença destas propriedades, criando funções com várias aplicabilidades distintas. Funções de resumo simples contêm apenas os três primeiros itens, e são utilizadas em vários âmbitos, em especial na estrutura de dados chamada de tabela de espalhamento. Por outro lado, é imposto para uma função de mão única que o cálculo de $\mathcal{H}^{-1}$ seja computacionalmente inviável, *i.e.* o item (iv) seja respeitado. Por fim, funções resistentes à segunda pré-imagem e/ou colisões são definidas como funções de resumo criptográfico, adequadas para utilização no contexto de segurança da informação.

Estas possibilitam a certeza da integridade de dados, mesmo que armazenados em um dispositivo inseguro. Intuitivamente, é desejável que não ocorra uma relação aparente entre entradas e saídas da função, considerando o resumo por completo e mesmo subpalavras deste. Outra característica desejada é o efeito avalanche, baseado no conceito de difusão (STALLINGS, 2010, pp. 72): trocar apenas um *bit* da mensagem m deve modificar cerca de metade dos *bits* do resumo, e vice-versa.

Adicionalmente, a relação da propriedade (vii) com funções de resumo é brevemente dirimida. Funções de compressão de mão única não admitem o título de funções de resumo, visto que a propriedade de compressão não é respeitada, embora seu nome leve a erro. Porém, são parametrizadas por uma chave, e utilizadas para a construção de cifras de bloco (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Algoritmo 9.25). Códigos de autenticação de mensagens, por outro lado, geralmente englobam todas as propriedades, pois funções de resumo criptográfico são utilizadas em sua construção, e a parametrização habilita autenticação da mensagem junto à integridade já fornecida.

Ademais, note que SEC e COL apresentam uma sutil diferença: na primeira, um adversário não pode escolher m_0, enquanto na segunda, quaisquer

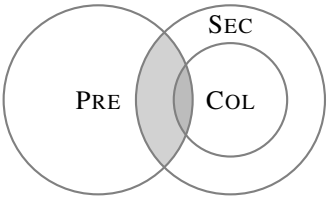

Figura 1 – Diagrama de Venn das resistências desejáveis para uma função de resumo no contexto de assinaturas digitais.

pares de mensagens podem ser testados. A resistência à colisões, portanto, implica na resistência à segunda pré-imagem, visto que basta um adversário fixar m_0 para simular o cômputo de m_1.

Na Figura 1, estão destacados os requisitos comuns para a utilização de funções de resumo no contexto de esquemas de assinatura digital, em vista da possibilidade de uma entidade maliciosa, geralmente chamada de adversário, desejar produzir assinaturas forjadas. Observe que, embora exista uma divisão estrita entre PRE e SEC, efetivamente, a segunda implica a primeira resistência (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Nota 9.20).

Enumeram-se algumas aplicações comuns para estas funções: a verificação da integridade de um arquivo, *i.e.* determinar se mudanças neste foram feitas ao longo de uma transmissão, ou qualquer outro evento; a fim de evitar o armazenamento de senhas em texto plano, mantêm-se apenas o resumo criptográfico destas, e no momento da autenticação do usuário perante o serviço, comparar apenas estes resumos[5]; resumos criptográficos são comumente empregados como identificadores únicos para um arquivo, *e.g. commits* em um sistema de controle de versões; entre outras aplicações, como a geração de números pseudoaleatórios.

2.2.1 Construção esponja

A construção esponja (BERTONI et al., 2011a), de característica iterativa, permite a generalização de funções de resumo, naturalmente com saídas de tamanho fixo, para funções com saídas de tamanho arbitrário, baseadas em uma função interna, geralmente uma permutação f de tamanho fixo b. Este valor, também chamado de largura, é composto da adição da taxa de *bits* r e da capacidade c. Assim, a construção opera em um estado de $b = r + c$ *bits*.

[5]Tabelas de resumos pré-computados, ou *rainbow tables*, são armazenadas a fim de atacar serviços que não empregam uma maneira mais elaborada de autenticação, *e.g.* um valor pseudoaleatório concatenado ao resumo criptográfico da senha do usuário.

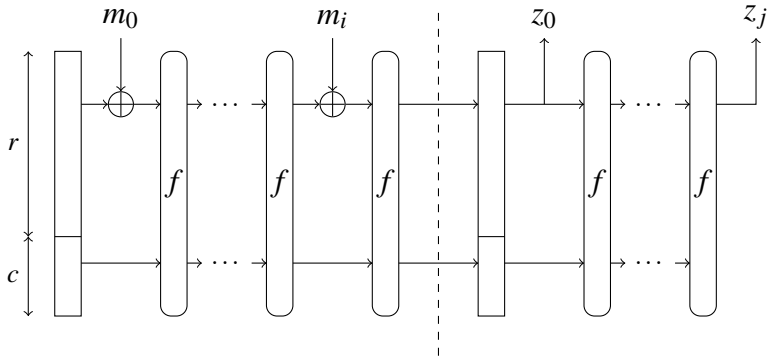

Figura 2 – Estrutura da construção esponja, onde $i, j \in \mathbb{N}^*$.

O estado inicial, análogo a um vetor de inicialização no contexto de algoritmos criptográficos, não necessita de valores especiais e é ocupado com valores nulos. A entrada m é preenchida com uma função de preenchimento `pad` de tal modo que $r \mid |m|$, e dividida em blocos de tamanho r. A fase de absorção de m pela esponja procede da seguinte maneira: a operação de ou exclusivo é calculada entre os blocos e os estados da construção, intercalados por aplicações de f.

Ao término do processamento dos blocos, a fase de compressão é iniciada, onde n blocos de tamanho r compõem a saída da função, novamente intercalados por aplicações de f, onde n é parametrizável pelo usuário. Os últimos *c bits* do estado nunca são diretamente afetados pelos blocos, e também nunca revelados durante a fase de compressão. Essencialmente, estão correlacionados com o nível de segurança da construção esponja. Assim, uma função esponja pode ser definida como $\text{SPONGE}[f, \texttt{pad}, r]$, e sua representação gráfica pode ser consultada na Figura 2, adaptada de (JEAN, 2016).

A função esponja KECCAK (BERTONI et al., 2011b) é definida a partir desta construção, e pode agir como uma função de resumo criptográfico. Existem sete permutações passíveis de utilização nesta função: defina $w = 2^\ell$, $\ell \in \{0, \ldots, 6\}$. Estas são chamadas de $\text{KECCAK} - f[b]$, onde $b = 25w$, cujo estado a é descrito como uma estrutura tridimensional com elementos em $\mathbb{F}_2$, de dimensões $5 \times 5 \times w$. Esta permutação é iterativa e consiste de um número de rodadas $n_R = 12 \times 2\ell$. Cada rodada R, por sua vez, consiste da composição de cinco etapas: $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$.

> *Etapa* θ. Calcula o ou exclusivo entre um elemento de a e todos os elementos das colunas adjacentes a este.

Etapa ρ. Dispersa os elementos entre cortes transversais verticais de a.

Etapa π. Rearranja elementos em cortes transversais horizontais de a.

Etapa χ. Modifica um elemento de uma linha de a de acordo com uma função não-linear de dois outros *bits* adjacentes. Análogo a uma caixa-S.

Etapa ι. Calcula o ou exclusivo entre o estado a e uma sequência gerada por um *linear-feedback shift register*[6] alimentado pelo índice da rodada atual, tornando a rodada assimétrica.

Tome `pad10*1` como uma função que gera palavras que iniciam e terminam com 1, e têm número não negativo de zeros. Formalmente, para uma mensagem qualquer m e um tamanho de saída $d \in \mathbb{N}^*$, a função esponja é definida como

$$\textsc{Keccak}[r,c](m,d) = \textsc{Sponge}[\textsc{Keccak} - f[r+c], \texttt{pad10*1}, r] \tag{2.9}$$

onde r tem um valor padrão de $1600-c$. Assim,

$$\textsc{Keccak}[c] = \textsc{Keccak}[1600-c, c]. \tag{2.10}$$

Finalmente, as funções padronizadas em (DWORKIN, 2015) como a família SHA-3 são definições de KECCAK com parâmetros fixos, *e.g.*

$$\text{SHA3-256}(m) = \textsc{Keccak}[512](m||01, 256). \tag{2.11}$$

2.3 ESQUEMAS DE ASSINATURA DIGITAL

Um esquema de assinatura digital é uma construção matemática que habilita a demonstração de certas propriedades sobre mensagens assinadas: nomeadamente, a autenticação do remetente, onde esta entidade pode ser facilmente identificada como a emissora da assinatura digital; a integridade da mensagem, *i.e.* a certeza de que esta não foi modificada ao ser transmitida por um canal possivelmente inseguro; e o não-repúdio do remetente, onde é impossível negar que uma mensagem foi assinada e enviada, após este fato.

Esquemas de assinatura digital são fortemente baseados em criptografia de chaves públicas, e consistem de três algoritmos: a geração de chaves

[6] Conjunto de registradores que deslocam *bits* a partir de funções lineares. Uma aplicação desta estrutura é a geração de números pseudoaleatórios.

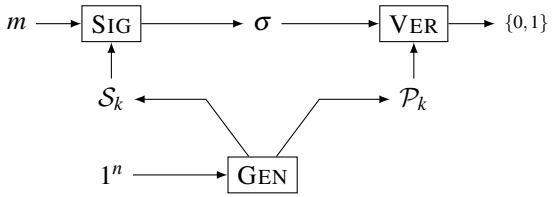

Figura 3 – Funcionamento típico de um esquema de assinatura digital.

GEN(1^n), que gera um par de chaves aleatório $(\mathcal{P}_k, \mathcal{S}_k)$ com parâmetro de segurança $n \in \mathbb{N}^*$; o algoritmo de assinatura SIG$(\mathcal{S}_k, m)$, que produz uma assinatura σ para uma mensagem m; e o algoritmo de verificação VER$(\mathcal{P}_k, m, \sigma)$, que retorna o estado de validade da assinatura como um valor verdade binário. De acordo com (GOLDREICH, 2004, Subseção 6.1.3), todas as assinaturas geradas por SIG devem ser verificáveis por VER utilizando todas as chaves geradas por GEN. Formalmente,

$$\forall (p,s) \in \text{GEN}(1^n), \forall w \in \{0,1\}^*, \\ \Pr[\text{VER}(p, w, \text{SIG}(s,w)) = 1] = 1. \quad (2.12)$$

É necessário apontar que esta definição não considera a segurança do esquema, visto que existem algoritmos construídos trivialmente que respeitam a equação acima, sem qualquer tipo de codificação a fim de ocultar mensagens. Na Figura 3, é apresentada uma visualização do comportamento de um esquema de assinatura digital genérico. Note que σ geralmente é composta da concatenação da mensagem original com a assinatura do resumo criptográfico desta, embora a saída do algoritmo SIG consista apenas da aplicação de uma função interna ao resumo.

Finalmente, a apresentação de um exemplo de esquema de assinatura digital é postergado para o Capítulo 3, onde serão explanados vários esquemas que, embora teoricamente similares, habilitam o entendimento conceitual de vários outros algoritmos com a mesma função, mas execuções distintas.

3 ASSINATURA DIGITAL BASEADA EM FUNÇÕES DE RESUMO CRIPTOGRÁFICO

Neste capítulo, os esquemas de assinatura digital baseados em funções de resumo criptográfico mais reconhecidos são apresentados, a fim de contextualizar a presença do esquema alvo, Winternitz, neste meio. Nomeadamente, a Seção 3.1 introduz o conceito de assinaturas únicas e fornece discussões sobre os esquemas LD-OTS, WOTS e WOTS+, respectivamente nas Subseções 3.1.1, 3.1.2 e 3.1.3. Adicionalmente, o esquema de poucas assinaturas HORS é explorado na Subseção 3.1.4. Por fim, os esquemas baseados em árvores de Merkle são definidos na Seção 3.2, e os exemplos MSS, XMSS e XMSS-MT são dirimidos nas Subseções 3.2.1, 3.2.2 e 3.2.3. Finalmente, a Seção 3.3 abrange uma visão cronológica dos esquemas discutidos ao longo deste Capítulo.

3.1 ESQUEMAS DE ASSINATURA ÚNICA

Esquemas de assinatura única, ou *one-time signature schemes*, possibilitam a assinatura de apenas uma mensagem, facilitando a falsificação deste processo no caso da reutilização do par de chaves. São construções fundamentais para a criação de esquemas baseados apenas em funções de resumo criptográfico — vários esquemas continuam utilizando assinatura única e compõem o estado da arte da literatura (BERNSTEIN et al., 2017; HüLSING et al., 2018).

3.1.1 LD-OTS

A utilização de funções de resumo criptográfico para a criação de esquemas de assinatura digital foi iniciada por Lamport (LAMPORT, 1979) e estendida subsequentemente em (DIFFIE; HELLMAN, 1976; MERKLE, 1989), permitindo assinar um *bit* de cada vez para uma dada mensagem. O par de chaves consiste de um par de palavras pseudoaleatórias x_0, x_1 como $\mathcal{S}_k$, e seus resumos como $\mathcal{P}_k$. O assinante assina um *bit* b distribuindo x_b, e o recipiente verifica se $\mathcal{H}(x_b)$ é o valor correto em $\mathcal{P}_k$. Como parte de $\mathcal{S}_k$ é distribuída, não é recomendável reutilizar o par de chaves, e os elementos não distribuídos devem ser destruídos. Assinar mensagens mais longas envolve o cálculo do resumo desta, visto que do contrário, o tamanho do par de chaves e assinaturas torna-se extremamente proibitivo.

Formalmente, tome um parâmetro de segurança $m \in \mathbb{N}$, geralmente considerado como o tamanho da saída da função de resumo criptográfico escolhido. Considere as funções[1] de mão única $f : \{0,1\}^m \longrightarrow \{0,1\}^m$ e de resumo criptográfico $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$. Seu funcionamento e algumas de suas características são discutidas abaixo.

Geração de chaves. Defina

$$\mathcal{S}_k = (x_{0,m-1}, x_{1,m-1}, \ldots, x_{0,0}, x_{1,0}) \xleftarrow{\$} \{0,1\}^{(m,2m)} \tag{3.1}$$

como a chave privada. A chave pública $\mathcal{P}_k$ é derivada de $\mathcal{S}_k$, computando $f(x)\ \forall x \in \mathcal{S}_k$. Logo,

$$\mathcal{P}_k = (f(x_{0,m-1}), f(x_{1,m-1}), \ldots, f(x_{0,0}), f(x_{1,0})) \tag{3.2}$$

$$= (y_{0,m-1}, y_{1,m-1}, \ldots, y_{0,0}, y_{1,0}). \tag{3.3}$$

Geração da assinatura. Tome uma mensagem M e calcule $d = \mathcal{H}(M)$. O resumo d pode ser representado como $d_{m-1} \ldots d_0$. A assinatura consiste de elementos de $\mathcal{S}_k$ operados por f de acordo com d:

$$\sigma = (f(x_{d_{m-1},m-1}), \ldots, f(x_{d_0,0})). \tag{3.4}$$

Verificação da assinatura. Para assegurar a corretude da assinatura σ, todos os blocos de σ precisam ser verificados separadamente através do recálculo de $d = d_{m-1} \ldots d_0$, para que os elementos corretos de $\mathcal{P}_k$ sejam escolhidos. Assim, σ está correta se

$$(\sigma_{m-1}, \ldots, \sigma_0) = (y_{d_{m-1},m-1}, \ldots, y_{d_0,0}). \tag{3.5}$$

É necessário comentar que assinaturas produzidas por este esquema podem ser forjadas (MERKLE, 1989, Capítulo 3), visto que é possível computar f para os *bits* da mensagem que são iguais a zero. Assim, uma soma de verificação representando a quantidade de zeros na mensagem original também é assinada e enviada ao recipiente, adicionando apenas $m \cdot \log_2(m)$ *bits* na assinatura. Esta característica será explicada em mais detalhes no próximo esquema.

Ademais, uma otimização simples para reduzir o tamanho de $\mathcal{S}_k$ pode ser implementada através da utilização de um gerador de números pseudoaleatórios criptograficamente seguro. Assim, é necessário apenas guardar uma palavra de tamanho m que age como a semente deste gerador. No caso de

[1] Na prática, o esquema pode ser instanciado utilizando $\mathcal{H}$ no lugar de f.

$\mathcal{P}_k$, basta guardar o resumo criptográfico de seus valores concatenados. Reduzir o tamanho das assinaturas exige soluções mais complexas; entretanto, a generalização deste esquema permite esta otimização.

3.1.2 WOTS

A introdução de um parâmetro que habilita compensar a criação de chaves e assinaturas menores com uma diminuição no desempenho do esquema foi sugerida por Winternitz e publicada por Merkle (MERKLE, 1989, Capítulo 5). Este esquema permite assinar múltiplos *bits* em cada bloco da assinatura, generalizando a proposta de Lamport. A definição do esquema é elaborada abaixo.

Um parâmetro de segurança m é necessário, como acima. O parâmetro Winternitz, da forma $w \in \mathbb{N}^* \setminus \{1\}$, também deve ser definido, que representa a quantidade de *bits* a serem assinados simultaneamente. A partir destes, são calculados os valores

$$t_1 = \left\lceil \frac{m}{w} \right\rceil, \; t_2 = \left\lceil \frac{\lfloor \log_2 t_1 \rfloor + 1 + w}{w} \right\rceil \text{ e } t = t_1 + t_2,$$

que representarão, respectivamente, a quantidade de palavras em base-w na assinatura referente à mensagem, à soma de verificação, e à quantidade total. Por fim, considere as funções de mão única $f : \{0,1\}^m \longrightarrow \{0,1\}^m$ e de resumo criptográfico $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$.

Geração de chaves. Defina

$$\mathcal{S}_k = (x_{t-1}, \ldots, x_0) \xleftarrow{\$} \{0,1\}^{(m,t)} \tag{3.6}$$

como a chave privada. A chave pública $\mathcal{P}_k = (y_{t-1}, \ldots, y_0)$ pode ser derivada de $\mathcal{S}_k$, computando $f^{2^w-1}(x) \; \forall x \in \mathcal{S}_k$. Logo,

$$\mathcal{P}_k = (f^{2^w-1}(x_{t-1}), \ldots, f^{2^w-1}(x_0)). \tag{3.7}$$

Geração da assinatura. Tome uma mensagem M e calcule $d = \mathcal{H}(M)$. Por conveniência, m ou w são escolhidos de forma que $w \mid m$, mas podem ser concatenados zeros ao resumo para que esta restrição seja satisfeita. O resumo d é dividido em uma t_1-tupla de palavras em base-w, $\mathcal{B}_1 = (b_{t-1}, \ldots, b_{t-t_1})$. Adicionalmente, uma soma de verificação é calculada usando a representação inteira dos elementos de $\mathcal{B}_1$: $c = \sum_{b \in \mathcal{B}_1} 2^w - 1 - b$. Novamente, c pode ser arredondado com zeros

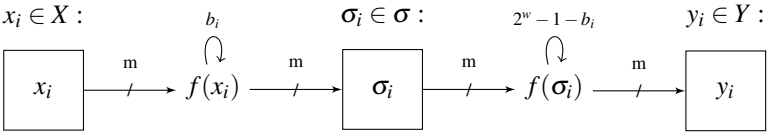

Figura 4 – Etapas de assinatura e verificação do WOTS, para $0 \leq i \leq t-1$.

até que $w \mid |c|$. Finalmente, c é dividido em uma t_2-tupla de palavras base-w, $\mathcal{B}_2 = (b_{t_2-1}, \ldots, b_0)$. Assim, $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ e

$$\sigma = (f^{b_{t-1}}(x_{t-1}), \ldots, f^{b_0}(x_0)). \tag{3.8}$$

Verificação da assinatura. Para assegurar a corretude da assinatura σ, todos os blocos σ_i precisam ser verificados separadamente através do cálculo das aplicações restantes de f. Assim, computando $\mathcal{B}$ novamente, σ está correta se

$$\mathcal{P}_k = (f^{2^w-1-b_{t-1}}(\sigma_{t-1}), \ldots, f^{2^w-1-b_0}(\sigma_0)). \tag{3.9}$$

Note que não é possível decrementar valores $b_i \in \mathcal{B}$, pois isto implicaria em achar uma pré-imagem de f, o que é computacionalmente inviável em vista de sua restrição imposta. Então, resta apenas a tentativa da falsificação de σ incrementando algum valor b_i. A soma de verificação c é necessária a fim de proteger o esquema contra esta situação. Observe que, em sua ausência, a criação de assinaturas σ' por uma entidade maliciosa é permitida, apenas através do cálculo de mais iterações de f em blocos da assinatura σ. Entretanto, com a presença de c, e consequentemente $\mathcal{B}_2$, ao modificar um valor $b_j \in \mathcal{B}_1$, para que o valor correto de $\mathcal{B}_2$ seja mantido, é necessário que uma pré-imagem de algum bloco $b_k \in \mathcal{B}_2$ seja calculada. Analogamente, modificar algum valor de $\mathcal{B}_2$ também tornará a verificação da assinatura impossível.

Diversas variantes do WOTS foram propostas com o intuito de reduzir requisitos de segurança, tamanhos de chaves e assinaturas, e tornar os processos de geração e verificação de assinaturas menos onerosos. Nomeadamente, para um ataque que consiste em ameaçar múltiplos usuários, cada qual com seu par de chaves, a possibilidade de sucesso aumenta de acordo com o subconjunto de instâncias almejado. Um esquema que combate especificamente este problema, chamado de WOTS-T, pode ser consultado em (HüLSING; RIJNEVELD; SONG, 2016).

Ademais, encontra-se em processo de padronização pela IETF outro esquema que combate este tipo de ataque, chamado WOTS-LM, publicado ori-

ginalmente como uma patente estadunidense (LEIGHTON; MICALI, 1995) — desta maneira buscando trazer a assinatura digital baseada em funções de resumo criptográfico para aplicações práticas. Em especial, para resolver o problema supracitado, os autores (MCGREW; CURCIO; FLUHRER, 2018) buscam introduzir identificadores únicos para cada instância do esquema, modificando essencialmente a estrutura da função f, assim combatendo ataques multiusuário.

3.1.3 WOTS+

O principal esquema da família Winternitz tem como ideia principal a reposição de f por uma família de funções $\mathcal{F}_k$, substituindo o processo de iterações sequenciais — utilizando apenas uma função — do esquema original (HüLSING; RAUSCH; BUCHMANN, 2013). Esta característica também permite uma prova de segurança mais simples, mas não menos poderosa, que desconsidera a necessidade da resistência à colisões de f e $\mathcal{H}$.

Defina os parâmetros m, w e a partir destes, t_1, t_2 e t, bem como a função de resumo criptográfico $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$ de maneira análoga ao esquema original. A família de funções que o WOTS+ utiliza é definida por

$$\mathcal{F}_m : \{f_k : \{0,1\}^m \longrightarrow \{0,1\}^m \mid k \in \mathcal{K}_m\},$$

ou seja, funções de mão única sem compressão, e $\mathcal{K}_m$ é interpretado como o espaço de chaves. A partir desta família, e de uma tupla de palavras pseudoaleatórias

$$r = (r_0, \ldots, r_{2^w-1}) \xleftarrow{\$} \{0,1\}^{(m,2^w)},$$

a nova função de iteração $c_k^i(x,r)$ é definida recursivamente como

$$\begin{aligned} c_k^0(x,r) &= x, \\ c_k^i(x,r) &= f_k(c_k^{i-1}(x,r) \oplus r_i). \end{aligned} \tag{3.10}$$

Geração de chaves. Defina $\mathcal{S}_k$ e r como tuplas de inteiros pseudoaleatórios, discutidos anteriormente. O algoritmo também deve escolher uma chave de função $k \xleftarrow{\$} \mathcal{K}$. Por fim,

$$\mathcal{P}_k = (c_k^{2^w-1}(x_{t-1}, r), \ldots, c_k^0(x,r)). \tag{3.11}$$

Geração da assinatura. Tome uma mensagem M, calcule o seu resumo $d = \mathcal{H}(M)$ e a soma de verificação c correspondente, produzindo $\mathcal{B}$. As considerações sobre preenchimento de m e c ainda se aplicam.

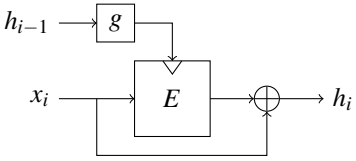

Figura 5 – Construção Matyas-Meyer-Oseas.

Aplicando a nova função de iteração, tem-se

$$\sigma = (c_k^{b_{t-1}}(x_{t-1}, r), \ldots, c_k^{b_0}(x_0, r)). \tag{3.12}$$

Verificação da assinatura. Novamente, é necessário calcular $\mathcal{B}$ para obter as iterações restantes até $\mathcal{P}_k$. Note que as primeiras b_i palavras de r não serão utilizadas. Assim, σ está correta se

$$\mathcal{P}_k = (c_k^{2^w-1-b_{t-1}}(\sigma_{t-1}, r), \ldots, c_k^{2^w-1-b_0}(\sigma_0, r)). \tag{3.13}$$

Note que a utilização de uma função conhecida para as iterações de f, como o AES, é factível. Porém, é necessário adaptá-la para que respeite as restrições impostas pelo esquema. Para que isto seja feito de maneira a manter um nível de segurança suficiente, a construção Matyas-Meyer-Oseas é empregada (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Algoritmo 9.41).

Assuma uma cifra de blocos genérica E_k parametrizada pela chave k, cujo bloco tem um tamanho de n *bits*. Tome um vetor de inicialização IV de tamanho n, e uma função g que adapte sua entrada para chaves do tamanho necessário por E_k. Deseja-se obter um resumo h_k de tamanho n que represente uma palavra x, e que sua inversibilidade seja computacionalmente inviável. Deste modo, $X = (x_0, \ldots, x_{k-1}), \forall x_i \in X, |x_i| = n$, ou seja, divida x em uma t-tupla cujos elementos têm tamanho n — preenchimento com zeros pode ser aplicado caso existam elementos que desrespeitem esta regra. O resultado deste processo, observado na Figura 5, adaptada de (JEAN, 2016), é definido recursivamente como

$$\begin{aligned} h_0 &= IV, \\ h_i &= E_{g(h_{i-1})}(x_i) \oplus x_i, 1 \leq i \leq t. \end{aligned} \tag{3.14}$$

Utilizando este método, é possível integrar o AES junto ao WOTS+, como forma de executar iterações para criar e verificar a assinatura, conforme sugerido em (HüLSING, 2013, Subseção 4.1), aproveitando implementações em

hardware de rotinas relacionadas, habilitando um grande ganho de desempenho no cômputo desta cifra.

3.1.4 HORS

Os esquemas discutidos até agora sofrem da mesma limitação: a exposição considerável de elementos relacionados diretamente com $\mathcal{S}_k$, habilitando a falsificação de assinaturas caso esta seja mantida. No caso de Lamport, metade dos valores são diretamente escolhidos como a assinatura, e em Winternitz, cadeias de resumos são diretamente computadas sobre todos os elementos de $\mathcal{S}_k$. Portanto, é desejável que esta exposição seja reduzida.

Para tal, a ideia de esquemas de "poucas assinaturas", ou *few-time signature schemes*, foi introduzida originalmente em (PERRIG, 2001) com o esquema *Bins and Balls*, uma aplicação do problema homônimo em probabilidade. Este trabalho é subsequentemente generalizado e otimizado em (REYZIN; REYZIN, 2002) na forma do esquema HORS, onde o conteúdo da mensagem controla a seleção de elementos em uma lista de inteiros trivial, construindo subconjuntos distintos, distribuídos como assinaturas.

De acordo com a definição original, tome os parâmetros $t, k \in \mathbb{N}, t \gg k$, e t como uma potência de 2. O tamanho dos pares de chaves e das assinaturas são fortemente relacionados com, respectivamente, t e k. O parâmetro de segurança $m = k \times \log_2 t$ garante que existam subconjuntos suficientes para todos os resumos possíveis. Seja o conjunto sequencial $T = \{0, \ldots, t-1\}$, a função de mão única $f : \{0,1\}^m \longrightarrow \{0,1\}^m$ e a função de resumo criptográfico $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$.

A função $\mathcal{H}$ utilizada deve apresentar também a característica de resiliência a subconjuntos de tamanho r, ou seja, deve ser computacionalmente inviável para um atacante, fornecidas r mensagens $(m_1, \ldots, m_r)$, encontrar $r+1$ mensagens tais que

$$\mathcal{H}(m_{r+1}) \subseteq \bigcup_{i=1}^{r} \mathcal{H}(m_i). \tag{3.15}$$

De outra forma, a probabilidade de que r assinaturas contenham valores que sejam mapeados para um resumo novo é desprezível. A escolha de r não é diretamente discutida, visto que uma função de resumo criptográfico usual provê esta característica para valores suficientemente pequenos (REYZIN; REYZIN, 2002, Apêndice A).

Geração de chaves. Defina $X = (x_{t-1}, \ldots, x_0) \xleftarrow{\$} \{0,1\}^{(m,t)}$, e adici-

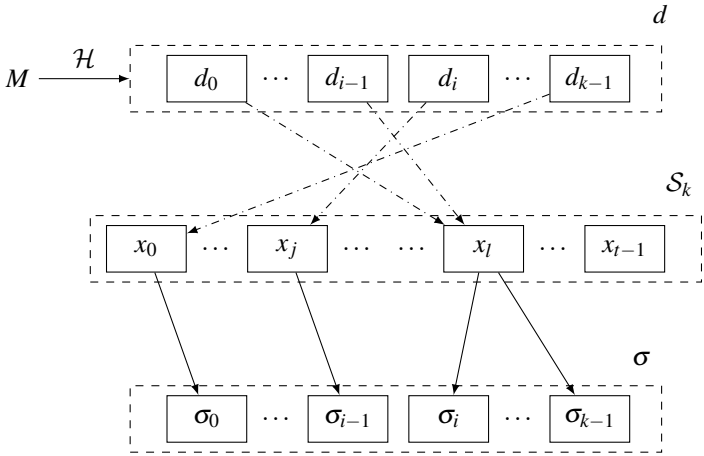

Figura 6 – Assinatura no HORS, para $0 \leq i \leq k-1$, $0 \leq j,l \leq t-1$.

one k para obter a chave privada. Portanto,

$$\mathcal{S}_k = (k, x_{t-1}, \ldots, x_0). \tag{3.16}$$

Aplique f em todos os elementos aleatórios para obter a chave pública, *i.e.*

$$\mathcal{P}_k = (k, f(x_{t-1}), \ldots, f(x_0)), \tag{3.17}$$

ou $\mathcal{P}_k = (k, y_{t-1}, \ldots, y_0)$.

Geração da assinatura. Tome uma mensagem M, calcule o seu resumo $d = \mathcal{H}(M)$, e represente-o como uma k-tupla de palavras em base-t, ou seja, $\mathbf{d} = (d_{k-1}, \ldots, d_0)$. A assinatura é composta dos elementos de $\mathcal{S}_k$ escolhidos pela representação inteira dos elementos de $\mathbf{d}$, ou seja,

$$\sigma = (x_{d_{k-1}}, \ldots, x_{d_0}). \tag{3.18}$$

Verificação da assinatura. Recalcule $\mathbf{d}$ a partir da mensagem M, e assegure a corretude da assinatura σ para cada bloco:

$$y_{d_i} = f(\sigma_i),\ 0 \leq i \leq k-1. \tag{3.19}$$

Parte deste processo é visualizado na Figura 6. Note que uma conversão de palavras binárias para inteiros está implícita no diagrama, como setas com preenchimento irregular, e que elementos d_i podem ter valores iguais,

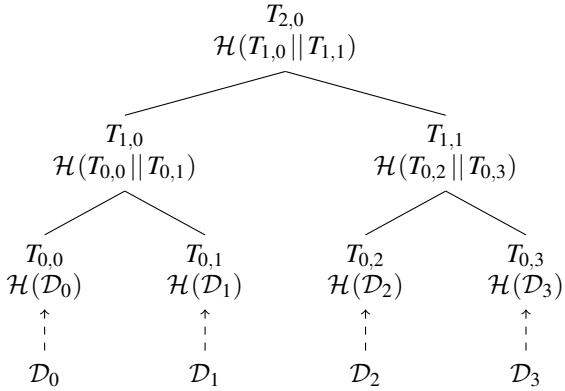

Figura 7 – Árvore de Merkle T, com altura $H = 2$.

e.g. $(d_0)_{10} = (d_{i-1})_{10} = l$. Percebe-se, a partir da descrição dos algoritmos, que o tamanho do par de chaves é proibitivo para aplicações práticas. Para tal, o algoritmo HORST é proposto em (BERNSTEIN et al., 2015, Capítulo 1), cuja definição faz uso das estruturas explicadas na Seção 3.2, a fim de reduzir substancialmente o tamanho de $\mathcal{P}_k$.

3.2 ESQUEMAS BASEADOS EM ÁRVORES DE MERKLE

A criação de um par de chaves para cada mensagem, bem como a infraestrutura necessária para relacionar múltiplas chaves a uma entidade, podem tornar-se processos extremamente onerosos. Desse modo, assim como esquemas de assinatura digital clássicos, é desejável que uma chave privada possa assinar múltiplos documentos. No contexto de esquemas baseados em funções de resumo criptográfico, a estrutura de dados chamada de árvore de Merkle, geralmente de característica binária e perfeita, pode ser aproveitada para tal.

De modo genérico, folhas de uma árvore de Merkle são construídas a partir do resumo criptográfico de dados que desejam ser inseridos nesta. Então, pais destas folhas computarão o resumo criptográfico do valor dos resumos de seus filhos concatenados, repetindo este processo até que a raiz seja preenchida, como na Figura 7. Considere uma árvore binária perfeita T de altura $H \in \mathbb{N}^* \setminus \{1\}$, onde um nodo é representado por

$$T_{h,j},\ 0 \leq h \leq H,\ 0 \leq j \leq 2^{H-h} - 1, \tag{3.20}$$

e uma função de resumo criptográfico $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$. As folhas são construídas a partir do resumo de qualquer dado $\mathcal{D}$, e seus nós intermediários a partir de seus filhos:

$$\begin{aligned} &T_{0,j} = \mathcal{H}(\mathcal{D}_j), \text{ e} \\ &T_{h,j} = \mathcal{H}(T_{h-1,2j} \,||\, T_{h-1,2j+1}),\ 1 \leq h \leq H. \end{aligned} \tag{3.21}$$

Esta estrutura também pode ser utilizada para verificação conjunta da integridade de múltiplos arquivos, *e.g.* em sistemas de arquivos, visto que qualquer mudança em um nó da árvore produzirá um valor distinto na raiz quando calculado novamente, descaracterizando a validade dos dados atrelados a esta.

Algoritmo 3 Funcionamento de TREEHASH$_h$.

Entrada: $h \in \{0,\ldots,2^H-1\}$, **S** ▷ índice de folha, pilha de nodos
Saída: **S** ▷ pilha atualizada
  LEAF ← LEAFCALC(i)
  enquanto LEAF.*height*() = **S**.*top*().*height*() **faça**
    TOP ← **S**.*pop*()
    LEAF ← $\mathcal{H}$(TOP||LEAF)
  fim enquanto
  S.*push*(LEAF)

Para calcular nodos de maneira eficiente, não é necessário armazenar T inteiramente. Considere a sub-rotina LEAFCALC$(k), 0 \leq k \leq 2^H-1$, que retorna o conteúdo da k-ésima folha, *i.e.* $\mathcal{H}(\text{OTS}^k_{\mathcal{P}_k})$, e um nodo $T_{\cdot,\cdot}$ com a rotina *height*(), que retorna sua altura em T. O algoritmo TREEHASH$_h$, munido de uma pilha **S** usual com as operações **S**.*pop*(), **S**.*push*($\cdot$) e **S**.*top*() calcula um nodo de altura h em 2^h chamadas de LEAF e 2^h-1 cálculos de $\mathcal{H}$. Este processo pode ser visualizado no Algoritmo 3.

Em qualquer momento de sua execução, o número máximo de nodos armazenados em **S** será h, chamados de nodos cauda, ou *tail nodes*. Ao seu estado final, **S** conterá apenas o nodo desejado na altura h. Note que a execução de TREEHASH repetidamente, como visto no Algoritmo 4, é realizada para calcular a raiz da árvore de Merkle $T_{H,0}$. Estas estratégias serão necessárias para a travessia em árvores de dimensões extensas, abordada na Subseção 3.2.2.

Algoritmo 4 TREEHASH para o cálculo de $T_{H,0}$.

Entrada: $H \geq 2$ ▷ altura da árvore T
Saída: $T_{H,0}$ ▷ nodo raiz
  para $i \leftarrow 0$ até $2^H - 1$ **faça**
    $\mathbf{S} \leftarrow \text{TREEHASH}_i(\mathbf{S})$
  fim para
  $T_{H,0} \leftarrow \mathbf{S}.pop()$

3.2.1 MSS

O primeiro esquema de assinatura digital baseado em árvores de Merkle (MERKLE, 1989, Capítulo 6) recomenda a assinatura de até 2^{20} mensagens. Informalmente, os dados inseridos em suas folhas são chaves públicas de diferentes instâncias de um esquema de assinatura única, como WOTS. Assim, ao construir a árvore de Merkle, a raiz será a chave pública deste esquema, que validará todas as folhas da árvore através de um caminho de autenticação. A chave privada pode ser descrita como o conjunto de chaves privadas das folhas, e a assinatura consiste da mensagem assinada pela instância do esquema localizado em uma folha qualquer, bem como todos os nós necessários para calcular o conteúdo da raiz da árvore.

Formalmente, seja uma árvore de Merkle T com $H \geq 2$. Tome um esquema de assinatura digital única genérico OTS, suas chaves privada e pública como $\text{OTS}_{\mathcal{S}_k}$ e $\text{OTS}_{\mathcal{P}_k}$, e os algoritmos de geração de chaves, geração de assinatura e verificação de assinatura como $\text{OTS}_{\mathcal{G}}, \text{OTS}_{\mathcal{S}}$ e $\text{OTS}_{\mathcal{V}}$. Por fim, considere uma função de resumo criptográfico $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$.

Geração de chaves. A chave privada é definida como uma 2^H-tupla de pares de chaves diferentes provenientes de OTS. Então, a partir de execuções de $\text{OTS}_{\mathcal{G}}$ para cada folha, tem-se

$$\mathcal{S}_k = ((\text{OTS}_{\mathcal{S}_k}^{2^H-1}, \text{OTS}_{\mathcal{P}_k}^{2^H-1}), \ldots, (\text{OTS}_{\mathcal{S}_k}^{0}, \text{OTS}_{\mathcal{P}_k}^{0})). \tag{3.22}$$

As folhas de T são preenchidas com as respectivas chaves públicas de OTS. Portanto, $T_{0,j} = \mathcal{H}(\text{OTS}_{\mathcal{P}_k}^{j})$, e T é construída recursivamente. Finalmente,

$$\mathcal{P}_k = T_{H,0}. \tag{3.23}$$

Geração da assinatura. Tome uma mensagem M e calcule o seu resumo $d = \mathcal{H}(M)$. Escolha uma folha de índice j não utilizada anteriormente e produza a assinatura OTS_{σ}^{j} a partir de d. O caminho de

autenticação AUTH é uma H-tupla necessária para recriar os resumos desejados a fim de compará-los com a raiz da árvore, e também deve ser incluído na assinatura. Assim, de acordo com (BERNSTEIN; BUCHMANN; DAHMEN, 2008, pp. 43),

$$\text{AUTH}_h = \begin{cases} T_{h,j/2^h-1}, \text{ se } \lfloor s/2^h \rfloor \equiv 1 \pmod 2, \\ T_{h,j/2^h+1}, \text{ se } \lfloor s/2^h \rfloor \equiv 0 \pmod 2, \end{cases} \tag{3.24}$$

para $0 \leq h \leq H-1$, representado visualmente na Figura 8. A assinatura final leva em conta todos estes itens. Portanto,

$$\sigma = (j, \text{OTS}^j_\sigma, \text{OTS}^j_{\mathcal{P}_k}, \text{AUTH}). \tag{3.25}$$

Verificação da assinatura. Para verificar uma assinatura perante a árvore T, é necessário primeiro verificar a assinatura única. Caso o resultado de $\text{OTS}_\mathcal{V}$ seja positivo, então os nodos do caminho de autenticação são reconstruídos através da definição recursiva abaixo:

$$\begin{aligned} &a_0 = \mathcal{H}(\text{OTS}^j_{\mathcal{P}_k}), \text{ e} \\ &a_h = \begin{cases} \mathcal{H}(a_{h-1} \,||\, \text{AUTH}_{h-1}), \text{ se } \lfloor j/2^{h-1} \rfloor \equiv 0 \pmod 2, \\ \mathcal{H}(\text{AUTH}_{h-1} \,||\, a_{h-1}), \text{ se } \lfloor j/2^{h-1} \rfloor \equiv 1 \pmod 2, \end{cases} \end{aligned} \tag{3.26}$$

para $1 \leq h \leq H$. Finalmente, se

$$\mathcal{P}_k = a_h, \tag{3.27}$$

a assinatura está válida.

Note que no passo de geração da assinatura, um índice da folha tem de ser escolhido, de modo que respeite uma restrição estrita: o nodo não pode ter sido selecionado anteriormente, a fim de prevenir a reutilização de uma instância de esquema de assinatura digital única. Para que essa escolha seja bem sucedida, é necessário que o esquema baseado em árvores de Merkle coordene quais índices foram utilizados.

Este conceito é chamado de gerenciamento de estado, e introduz vários obstáculos para o funcionamento do esquema (MCGREW et al., 2016), como a necessidade de salvar frequentemente informações sobre o estado em armazenamento não volátil. Ademais, este é geralmente codificado em $\mathcal{S}_k$, e portanto, duas mensagens não podem ser assinadas com a mesma chave privada, criando problemas de sincronização de chaves entre múltiplos dispositivos. Esta restrição é removida em (GOLDREICH, 2004, Construção

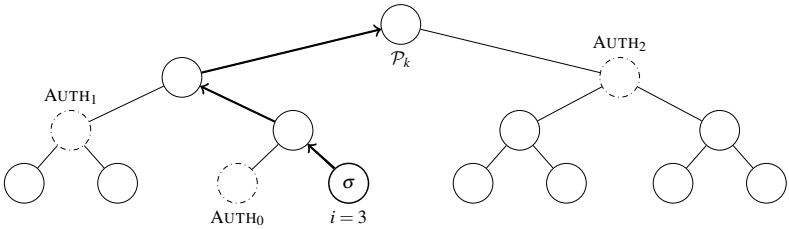

Figura 8 – Caminho de autenticação para a folha de índice $i = 3$.

6.4.16), onde o esquema deve escolher um nodo aleatoriamente, assim introduzindo a probabilidade de que um nodo possa ser escolhido duas vezes, relativa à quantidade de folhas. Esquemas que lidarão com esta característica serão discutidos futuramente.

Observe que o tamanho da chave pública é extremamente proibitivo e escala com o número de folhas da árvore. Para resolver esta limitação, um novo parâmetro é adicionado ao esquema, na forma de um gerador de números pseudoaleatórios determinístico, cuja semente será a nova chave privada (GOLDREICH, 2004, Subseção 6.4.2.3). Adicionalmente, este recurso implica que a árvore não necessita ser construída totalmente, posto que a assinatura de novas mensagens implica na criação dinâmica de pares de chaves para estas instâncias de esquemas de assinatura única. Caso este gerador apresente como característica o sigilo encaminhado, ou *forward secrecy*, onde valores gerados no passado não são ameaçados caso a função seja comprometida, o esquema também será caracterizado como tal (BERNSTEIN; BUCHMANN; DAHMEN, 2008, pp. 45): chaves privadas comprometidas não implicarão na falsificação de mensagens assinadas com chaves antigas.

Note que uma árvore de altura razoável para utilização em aplicações práticas tem seu armazenamento total impossibilitado na maioria absoluta dos dispositivos alvo, em vista da quantidade excessiva de nodos. A fim de reduzir este ônus, estruturas virtuais são empregadas, onde apenas partes relevantes da árvore de Merkle em uso estão acomodadas em memória. Entretanto, é necessário que quaisquer nodos possam ser autenticados de maneira modesta, levando em conta restrições de processamento e armazenamento. Assim, um problema de travessia entre os nodos da árvore de Merkle deve ser resolvido, e a utilização de algoritmos otimizados é crucial na execução de alguns esquemas, visto que a extensão das árvores de assinaturas é um fator desejável na criação de novos esquemas. Esta característica será discutida na Subseção 3.2.3.1.

Finalmente, é razoável destacar que a função $\mathcal{H}$ utilizada neste es-

quema tem a restrição implícita de ser resistente à colisões, visto que a modificação de qualquer nodo na árvore sem que ocorra uma divergência na raiz implica em uma colisão de resumos. Este é notoriamente o requisito de segurança mais explorado no contexto de criptoanálise aplicada sobre funções de resumo criptográfico, *e.g.* colisões encontradas em várias funções baseadas na estrutura Merkle—Damgård. Assim, é razoável construir esquemas com menos requisitos, tornando-os naturalmente mais seguros.

Um esquema baseado em árvores de Merkle que necessita apenas de resistência à segunda pré-imagem em todos os seus componentes, chamado de SPR-MSS, é proposto em (DAHMEN et al., 2008). Consiste na reestruturação das folhas da árvore, onde o resumo da concatenação dos elementos de cada chave pública originalmente calculado é substituído por uma reorganização destes em uma árvore possivelmente desbalanceada. Posteriormente, máscaras de *bits* aleatórias são aplicadas nestes elementos, e também em cada criação de nodo na árvore de Merkle. Esta construção será descrita formalmente junto ao esquema a seguir.

3.2.2 XMSS

A redução dos requisitos de segurança, bem como a presença do sigilo encaminhado, são as principais características do esquema XMSS. Estes recursos foram explorados em (BUCHMANN; DAHMEN; HüLSING, 2011), e culminam em uma forma mais eficiente de aplicar as ideias impostas em SPR-MSS, junto à utilização de um gerador de números pseudoaleatórios de funcionamento particular. De forma complementar, requisitos de armazenamento de assinaturas são diminuídos através do uso de algoritmos de travessia.

A descrição original deste esquema utiliza instâncias de um esquema de assinatura única baseado em Winternitz chamado de WOTS-PR, criado com o intuito de possuir requisitos de segurança mínimos a fim de manter a necessidade de uma função de resumo apenas resistente à segunda pré-imagem. Entretanto, de acordo com (LAFRANCE, 2017, Seção 4.2), a prova de segurança para este esquema apresenta erros. Portanto, a descrição a seguir tomará o esquema de assinatura única utilizado como o WOTS+, cuja prova de segurança não contém erros conhecidos.

Assim, tome um parâmetro de segurança m e o parâmetro de Winternitz w, a família de funções

$$\mathcal{F}_m : \{f_k : \{0,1\}^m \longrightarrow \{0,1\}^m \mid k \in \mathcal{K}_m\},$$

uma árvore de Merkle T com altura H e uma família de funções de resumo

$$\mathcal{H}_m : \{h_k : \{0,1\}^{2m} \longrightarrow \{0,1\}^m \mid k \in \mathcal{K}_m\}.$$

Um gerador de números pseudoaleatórios é construído a partir de f a fim de assegurar sigilo encaminhado, definido como

$$\text{PRF}_a(b) = f_a(1) \,||\, \ldots \,||\, f_a(b),\ a \in \mathbb{N},\ b \in \{0,1\}^m. \tag{3.28}$$

O esquema de assinatura única OTS é definido como WOTS+.

Observe que, por conta de sua semelhança com o esquema SPR-MSS, a estrutura da árvore de Merkle é modificada. A função de construção da árvore é remodelada a fim de utilizar máscaras de *bits* $r_i \xleftarrow{\$} \{0,1\}^{2m}$ para cada nível da árvore, ou seja,

$$T_{h,j} = h_k(r_i \oplus (T_{h-1,2j} \,||\, T_{h-1,2j+1})),\ 1 \leq h \leq H,\ 0 \leq j \leq 2^{H-h} - 1. \tag{3.29}$$

O conteúdo das folhas, originalmente o resumo criptográfico de elementos concatenados de $\text{OTS}_{\mathcal{P}_k}$, é substituído por uma estrutura chamada de árvore-l, construída a partir dos elementos desta chave de forma similar à T, mas com máscaras iguais para todas as árvores referentes aos esquemas de assinatura única. Note que o número de elementos de $\text{OTS}_{\mathcal{P}_k}$, o inteiro t em esquemas baseados em Winternitz, pode não ser uma potência de 2, e portanto a árvore-l deve ser balanceada tal que qualquer nodo sem irmão à direita é movido para um nível mais alto, de forma a tornar-se um irmão à direita de outro nodo. Assim, a altura total é de $\lceil \log_2 t \rceil$, mas não considerada no contexto de T.

Geração de chaves. Tome uma semente $s \xleftarrow{\$} \{0,1\}^m$ e uma função de resumo $h_k \xleftarrow{\$} \mathcal{H}_m$. Gere o conjunto de máscaras

$$\mathbf{r} = (r_0, \ldots, r_{H+\lceil \log_2 t \rceil - 1}) \xleftarrow{\$} \{0,1\}^{(2m, H+\lceil \log_2 t \rceil)}. \tag{3.30}$$

Para gerar as chaves privadas das instâncias de esquemas de assinatura única, defina

$$\text{OTS}^i_{\mathcal{S}_k} = \text{PRF}_t(f_s(i)),\ \forall i \in \{0, \ldots, 2^H - 1\}. \tag{3.31}$$

A árvore T é construída utilizando a compressão de $\text{OTS}^i_{\mathcal{P}_k}$ supracitada, bem como o processo usual para o cálculo de resumos intermediários. Finalmente, a chave privada deve guardar o estado do esquema, que pode ser representado como o índice da primeira folha não utilizada.

Portanto,

$$\mathcal{S}_k = (s, 0), \tag{3.32}$$

e naturalmente, a chave pública deve conter as máscaras utilizadas em T, *i.e.*

$$\mathcal{P}_k = (T_{H,0}, \mathbf{r}). \tag{3.33}$$

Geração da assinatura. A assinatura é construída de maneira muito similar a esquemas de Merkle clássicos. Para uma mensagem M, um resumo $d = h_k(M)$, e um índice j não utilizado anteriormente, OTS^j_σ é calculada a partir de d. O caminho de autenticação AUTH é construído com o auxílio de algoritmos de travessia. Assim,

$$\sigma = (j, \text{OTS}^j_\sigma, \text{AUTH}). \tag{3.34}$$

Verificação da assinatura. Para verificar uma assinatura perante T, assegure que $\text{OTS}_\mathcal{V}(\text{OTS}^j_{\mathcal{P}_k}, \text{OTS}^j_\sigma) = 1$, e utilize esta chave para reconstruir a árvore-l com as máscaras $\mathbf{r}$, obtendo a raiz desta. Junto a AUTH, compute o caminho de nodos $(a_0, \ldots, a_h)$. Finalmente, se

$$\mathcal{P}_k = a_h, \tag{3.35}$$

a assinatura está válida.

O esquema descrito tem estrutura similar à variante em processo de padronização pela IETF, que faz uso de WOTS+ em suas folhas. Este esquema também conta com outros arcabouços que permitem a resistência a ataques específicos, *e.g.* a associação da assinatura com o usuário e a posição na árvore impede ataques que almejam forjar assinaturas para um grande número de instâncias de esquemas de assinatura única (HüLSING et al., 2018, Seção 9.1). Entretanto, seu funcionamento geral é similar a esquemas clássicos, e o gerenciamento de estado é mantido. As otimizações apresentadas futuramente são implementadas e testadas principalmente neste esquema.

3.2.3 XMSS-MT

Uma das principais limitações dos esquemas baseados em árvores de Merkle é a quantidade de assinaturas a serem conectadas com uma árvore, ou chave pública. Para solucionar este problema, o conceito de hiper-árvores é proposto em diversos trabalhos, com o intuito de encadear camadas de árvores de Merkle, conectadas por uma instância de esquema de assinatura única, de tal forma que a folha da árvore superior assina a raiz da árvore inferior.

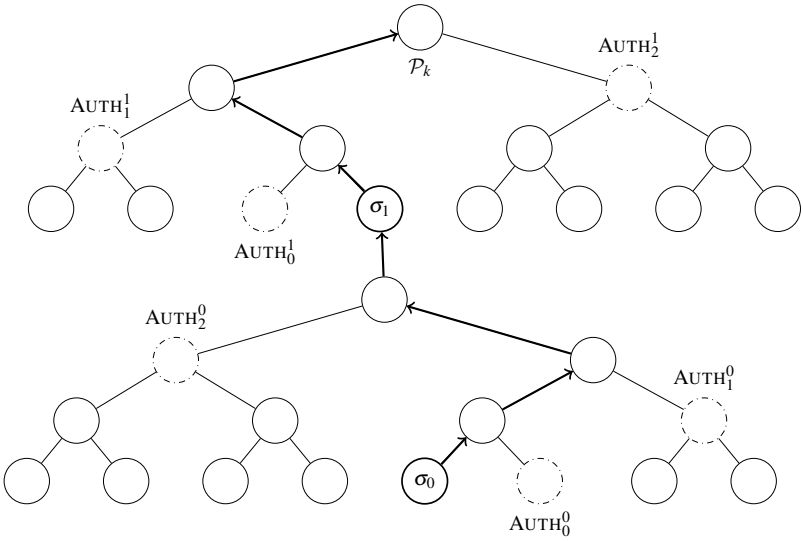

Figura 9 – Caminho de autenticação para a folha $j = 3 \cdot 2^{h_0} + 5$, em uma hiper-árvore com $d = 2, h_0 = h_1 = 3$.

A ideia original é introduzida em (BUCHMANN et al., 2006) na forma do esquema CMSS, que habilita a assinatura de até 2^{40} mensagens com um par de chaves, relacionado a uma estrutura virtual de duas camadas, onde apenas uma árvore na parte inferior é criada de cada vez a fim de reduzir requisitos de armazenamento. Posteriormente, este esquema é generalizado em (BUCHMANN et al., 2007) como GMSS, que utiliza um número arbitrário de camadas de árvores, com alturas e parâmetros de esquemas de assinatura única diferentes para cada uma destas, atingindo o número de 2^{80} mensagens por par de chaves.

Naturalmente, este recurso é aplicado ao esquema XMSS a fim de produzir a variante XMSS+ (HüLSING; BUSOLD; BUCHMANN, 2012), com uma árvore superior e inferior, e posteriormente, a versão XMSS-MT (HüLSING; RAUSCH; BUCHMANN, 2013), generalizada para qualquer número de camadas de árvores, abordada abaixo. Para descrevê-la completamente, é necessário apontar que algoritmos de travessia são essenciais no funcionamento prático de esquemas com hiper-árvores, visto que estas estruturas existirão apenas parcialmente em qualquer dispositivo, embora nodos em várias partes da hiper-árvore são necessários para construir caminhos de autenticação. Deste modo, o algoritmo BDS (BUCHMANN; DAHMEN; SCHNEIDER, 2008) é exposto junto ao esquema, a fim de demonstrar a complexidade

real do gerenciamento de estado atrelado a este tipo de construção.

3.2.3.1 Travessia de árvores de Merkle BDS

O algoritmo admite quatro entradas, na forma da altura H da árvore T, um índice de folha $s \in \{0, \ldots, 2^H - 2\}$, e um parâmetro

$$K \in \mathbb{N}^* \setminus \{1\},\ K < H,\ {}^{H-K}/_{2} \equiv 0 \pmod 2 \tag{3.36}$$

que representa uma compensação entre armazenamento e tempo de execução do algoritmo, permitindo que uma execução total de TREEHASH, ou seja, para todas as $2^H - 1$ folhas, seja substituída por ${}^{H-K}/_{2}$ instâncias desta rotina. Finalmente, o estado $\text{STATE}_{\text{BDS}}$ é composto das seguintes estruturas:

(i) AUTH, lista de tamanho H que armazena o caminho de autenticação atual;

(ii) KEEP, lista de tamanho $H-1$ que armazena nodos úteis para o cálculo de nodos de autenticação à esquerda;

(iii) $\{\text{RETAIN}_h : H - K \leq h \leq H - 2\}$, pilhas com nodos à direita perto da raiz;

(iv) **S**, pilha munida das operações usuais;

(v) $\{\text{TREEHASH}_h : 0 \leq h \leq H - 3\}$, *i.e.* várias instâncias do Algoritmo 3.

As listas são munidas da operação $remove(\cdot)$, cujo argumento representa um índice válido de acordo com o tamanho da lista. As instâncias TREEHASH_h compartilham **S**. São inicializadas por $init(\cdot)$, cujo argumento é um índice de folha que criará o nodo LEAF no Algoritmo 3, e podem ser executadas totalmente por $update()$. Ademais, são munidas da função $height()$, que retorna a menor altura entre nodos localizados nesta instância. Em casos anômalos, se **S** estiver vazia, $\text{TREEHASH}_h.height() = h$, e se esta instância não tiver sido inicializada ou já tiver sido terminada, $\text{TREEHASH}_h = \infty$.

O algoritmo retorna o caminho de autenticação para a folha $T_{0,s+1}$. É inicializado na etapa de geração de chaves do esquema. AUTH recebe primeiramente o caminho de autenticação para a folha $T_{0,0}$, representado por $\text{AUTH} = (T_{0,1}, \ldots, T_{H-1,1})$. As instâncias TREEHASH_h têm como nodos iniciais $T_{h,3}$. As pilhas RETAIN_h recebem os nodos $T_{h,2k+3}$, $2^{H-h-1} - 2 \leq k \leq 0$. Execuções posteriores do algoritmo mantém $\text{STATE}_{\text{BDS}}$ preparado anteriormente.

O cálculo de nodos intermediários depende se este está à esquerda ou à direita. A descrição detalhada de BDS, no Algoritmo 5, contempla ambos os casos, e será discutida em detalhes. Para uma folha $T_{0,j}$, τ é a altura do primeiro pai à esquerda de $T_{0,j}$. Este valor codifica as alturas nas quais o caminho de autenticação para $T_{0,j+1}$ necessita de novos nodos. Nodos à direita são necessários para alturas até $\tau - 1$, e um nodo à esquerda na altura τ. Caso este nodo esteja à esquerda, então $\tau = 0$, e do contrário, $\tau = \max(\{2^h \mid j+1 : 1 \leq h \leq H\})$.

Algoritmo 5 Travessia de árvores de Merkle BDS.

Entrada: $H \geq 2$, K (Eq. 3.36), $s \in 0,\ldots,2^H-2$, $\text{STATE}_{\text{BDS}}$ ▷ altura da árvore T, parâmetro de compensação, índice de folha, estado do algoritmo
Saída: AUTH^{s+1} ▷ caminho de autenticação para a folha $s+1$

```
τ ← max({2^h | s+1 : h ∈ 1,...,H})
se ⌊s/2^(τ+1)⌋ ≡ 0 (mod 2) e τ < H−1 então
    KEEP_τ ← AUTH_τ
fim se
se τ = 0 então
    AUTH_0 ← LEAFCALC(s)
senão se τ > 0 então
    AUTH_τ ← H(AUTH_(τ−1) || KEEP_(τ−1))
    KEEP.remove(τ−1)
    para h = 0 até τ−1 faça
        se h < H−K então
            AUTH_h ← TREEHASH_h.pop()
        senão
            AUTH_h ← RETAIN_h.pop()
        fim se
    fim para
    para h = 0 até min(τ−1, H−K−1) faça
        se s+1+3·2^h < 2^H então
            TREEHASH_h.init(s+1+3·2^h)
        fim se
    fim para
fim se
para i = 0 até (H−K)/2 faça
    k ← min({TREEHASH_j.height() : j ∈ 0,...,H−K−1})
    TREEHASH_k.update()
fim para
```

Para computar os nodos à esquerda, a utilização da lista KEEP será

crucial para calcular nodos de autenticação à esquerda utilizando apenas um cálculo de $\mathcal{H}$. Observe que, se o pai da folha $T_{0,s}$ na altura $\tau+1$ é um nodo à esquerda, então AUTH_τ está à direita e pode ser armazenado em KEEP_τ. Caso $\tau = 0$, então a folha está à esquerda e servirá como parte do caminho de autenticação para sua irmã. Portanto, AUTH_0 recebe $\text{LEAFCALC}(s)$. Do contrário, a folha está à direita, e o cálculo de AUTH_τ é necessário. Neste caso, $\text{AUTH}_{\tau-1}$ deve existir e ser seu filho à esquerda. Resta buscar o filho à direita, que estará armazenado em $\text{KEEP}_{\tau-1}$. Portanto, AUTH_τ pode ser calculado. Uma explicação detalhada deste processo pode ser encontrada em (BUCHMANN; DAHMEN; SCHNEIDER, 2008, Seção 2.1).

O cômputo de nodos à direita é feito desde as folhas, visto que nenhum dos seus nodos filho foram utilizados em caminhos de autenticação anteriores. O parâmetro K é empregado nesta parte do algoritmo. Nodos cuja altura estão em $H-K \leq h \leq H-2$ são retirados das pilhas RETAIN_h e movidos para AUTH_h. Nodos com altura $h < H-K$ fazem uso dos nodos armazenados nas instâncias TREEHASH_h, então subsequentemente inicializadas com os nodos $s+1+3\cdot 2^h$, se necessário. Finalmente, algumas instâncias cujos nodos têm menor altura são atualizadas, de acordo com o critério em (BERNSTEIN; BUCHMANN; DAHMEN, 2008, Seção 4.5), a fim de adicionar nodos relevantes ao estado do algoritmo.

3.2.3.2 Descrição do esquema

Tome o parâmetro de segurança m, a família de funções $\mathcal{F}_m$, uma função de resumo $\mathcal{H} : \{0,1\}^{2m} \longrightarrow \{0,1\}^m$, o esquema de assinatura única OTS definido como WOTS+, o número de camadas da hiper-árvore $d \in \mathbb{N}^*$, e para cada camada $0 \leq i \leq d-1$, a altura da árvore h_i, o parâmetro de Winternitz w_i, e consequentemente t_i, e o parâmetro BDS k_i. Novamente, um gerador de números pseudoaleatórios PRF é construído a partir de f.

Geração de chaves. Calcule o valor

$$k = \max(\{h_i + \lceil \log_2 t_i \rceil : 0 \leq i \leq d-1\}), \tag{3.37}$$

e gere uma k-tupla de máscaras de *bits* aleatórias $\mathbf{r}$, reutilizadas em cada camada da árvore. Uma d-tupla de sementes para cada camada da hiper-árvore também é gerada desta maneira, e estados do algoritmo BDS são inicializados: $\text{STATE} = (\text{STATE}^0_{\text{BDS}}, \ldots, \text{STATE}^{d-1}_{\text{BDS}})$. Utilizando instâncias do algoritmo TREEHASH e a tupla $\mathbf{s}$ a fim de criar os pares de chaves de WOTS+, os valores $T_{h_i,0},\ 0 \leq i \leq d-2$ são calculados e assinados pela primeira folha da árvore superior. Estas assinaturas

são então extraídas como $\Sigma = (\sigma_1^T, \ldots, \sigma_{d-1}^T)$.

Note que apenas as assinaturas das primeiras raízes à esquerda são armazenadas, visto que será a primeira árvore a ser utilizada para assinar mensagens. Para facilitar a construção de futuros caminhos de autenticação e árvores intermediárias, sementes pseudoaleatórias, estados do algoritmo BDS e instâncias do algoritmo TREEHASH são também armazenados em uma estrutura

$$\text{NEXT} = \big(s_0^n, \ldots, s_{d-2}^n, \text{TREEHASH}_0^n, \ldots, \text{TREEHASH}_{d-2}^n, \text{STATE}_{\text{BDS}}^{0,n}, \ldots, \text{STATE}_{\text{BDS}}^{d-2,n}\big), \quad (3.38)$$

onde n denota a próxima árvore a ser utilizada na camada i. Finalmente,

$$\mathcal{S}_k = (\mathbf{s}, \text{STATE}, \Sigma, \text{NEXT}), \quad (3.39)$$

$$\mathcal{P}_k = (T_{h_{d-1},0}, \mathbf{r}). \quad (3.40)$$

Geração da assinatura. Para uma mensagem M e resumo $d = \mathcal{H}(M)$, e um índice j não utilizado anteriormente, calcula-se OTS_σ^j. Os caminhos de autenticação são retirados de STATE, e as assinaturas intermediárias em Σ. Assim, a assinatura é composta de

$$\sigma = (j, \text{OTS}_\sigma^j, \Sigma, \text{AUTH}^0, \ldots, \text{AUTH}^{d-1}). \quad (3.41)$$

A árvore deve ser então inteiramente atualizada através dos algoritmos TREEHASH e BDS, modificando os estados e tornando a próxima folha disponível para uso. Em especial, caso a última folha de uma árvore for utilizada, isto significa que uma nova árvore deve ser criada. Assim, a chave privada deve ser atualizada com novos elementos $\mathbf{s}', \text{STATE}', \Sigma', \text{NEXT}'$. Mais detalhes sobre este processo podem ser encontrados em (HüLSING; RAUSCH; BUCHMANN, 2013, Capítulo 2).

Verificação da assinatura. Para verificar σ perante à hiper-árvore, OTS_σ^j é aferida a fim de construir $\text{OTS}_{\mathcal{P}_k}$. Com este nodo e AUTH^0, a raiz da árvore é construída, a fim de verificar a primeira assinatura em Σ. Este processo é repetido $d-1$ vezes, utilizando todos os caminhos de autenticação e assinaturas em Σ fornecidos por σ, e a saída da última iteração é comparada com $T_{h_{d-1},0}$. A assinatura é válida se e somente se estes valores forem iguais. Este processo pode ser visualizado na Figura 9.

Adicionalmente, como mencionado na Subseção 3.1.2, o esquema

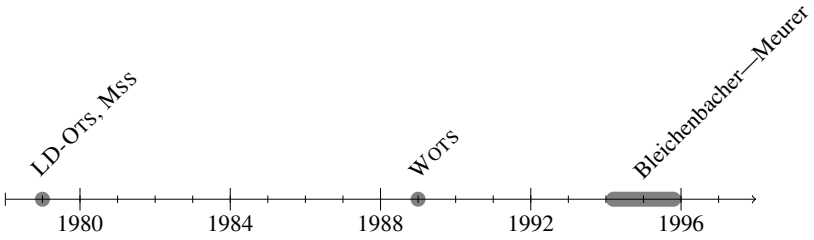

Figura 10 – Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1979 até 1997.

XMSS-T é definido em (HüLSING; RIJNEVELD; SONG, 2016), como uma variante do XMSS-MT resistente à ataques multiusuário. Isto é realizado através da atribuição de um endereço único à cada nodo na árvore, e derivando deste um resumo criptográfico calculado por funções com novas noções de resistência à pré-imagem, segunda pré-imagem e colisões, a fim de reduzir as ameaças ocasionadas por este tipo de ataque.

3.3 VISÃO TEMPORAL

Vários esquemas foram citados ao longo das seções anteriores, provendo uma contextualização teórica para o estudo aprofundado de variantes seletas. Deste modo, é razoável visualizar de maneira cronológica a evolução do estudo de assinaturas digitais baseadas em funções de resumo criptográfico. De acordo com a Figura 10, nas duas décadas iniciais, esta área é estabelecida pela pesquisa em (LAMPORT, 1979; MERKLE, 1979), definindo os esquemas seminais LD-OTS e MSS, e no meio deste período, o esquema WOTS é publicado em (MERKLE, 1989). Resultados teóricos e otimizações para o esquema de Lamport também fazem parte de publicações com destaque, *e.g.* (BLEICHENBACHER; MAURER, 1994), que formaliza uma representação de esquemas de assinatura única como grafos acíclicos dirigidos.

Entretanto, as últimas duas décadas de pesquisa culminaram em ideias utilizadas por esquemas considerados como o estado da arte. Os conceitos de esquemas de poucas assinaturas, redução de requisitos de segurança para resistência à segunda pré-imagem, hiper-árvores e resistência a ataques multiusuário, todos no contexto de assinaturas digitais com funções de resumo criptográfico, podem ser mapeados para este período. As Figuras 10 e 11 contemplam todos os esquemas citados nas Seções 3.1 e 3.2, à exceção da

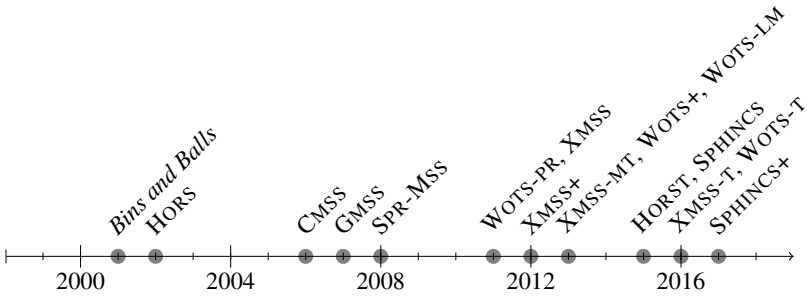

Figura 11 – Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1998 até 2018.

família SPHINCS (BERNSTEIN et al., 2015; BERNSTEIN et al., 2017), que renuncia o gerenciamento de estado, característica presente em todos os esquemas baseados em árvores de Merkle aprofundados anteriormente.

Esta família utiliza várias estratégias combinadas a fim de evitar esta característica: a altura da hiper-árvore é aumentada substancialmente, para que o número de assinaturas que podem ser feitas com apenas um par de chaves seja maior; nos nodos folha mais inferiores, o esquema HORST é utilizado para evitar colisões entre assinaturas; o índice de folha escolhido depende de um valor pseudoaleatório; entre outros recursos. Entretanto, as otimizações descritas no Capítulo 4 têm aplicação reduzida a estes esquemas, visto que o cerne do processamento total destes não é causado pela utilização de instâncias Winternitz, restritas ao cálculo de assinaturas entre árvores intermediárias.

4 OTIMIZAÇÃO DO ESQUEMA WINTERNITZ

Neste capítulo, duas otimizações são apresentadas para a família de esquemas Winternitz. A primeira, definida na Seção 4.1, consiste de uma modificação em *bits* inutilizados na soma de verificação calculada ao longo do processo de assinatura, tornando a verificação desta menos onerosa. A segunda, definida na Seção 4.2, apresenta uma estratégia para modificar a entrada do algoritmo, a fim de torná-la adequada para o cômputo eficiente das etapas de geração ou verificação de uma assinatura. Consequências destas modificações são discutidas na Seção 4.3, e o impacto em esquemas da família XMSS é apresentado na Subseção 4.3.1.

4.1 WOTS-B

A primeira variante de WOTS proposta é baseada na alteração da estrutura da soma de verificação, explicada na Subseção 3.1.2, cujo cálculo é essencial para o impedimento de falsificação trivial de assinaturas por meio da aplicação de f sobre cadeias de resumos pré-existentes. Esta estratégia é abordada de maneira diferente no esquema WOTS-LM, onde um de seus parâmetros comanda a quantidade de deslocamentos à esquerda realizados sobre a representação binária desta soma (MCGREW; CURCIO; FLUHRER, 2018, Seção 4.4).

Entretanto, caso este parâmetro seja escolhido de maneira inábil, a etapa de verificação será executada de forma menos eficaz em relação ao tempo de execução. Ademais, é possível calcular o número de *bits* não utilizados na soma de verificação de maneira exata. De acordo com a Subseção 3.1.2, considere a t_1-tupla $\mathcal{B}_1$ como os valores do resumo da mensagem divididos em palavras de base-w, e a soma de verificação $c = \sum_{b \in \mathcal{B}_1} 2^w - 1 - b$. Seu valor máximo é definido por

$$c_{\max} = \sum_{i=t-t_1}^{t-1} (2^w - 1 - 0) = t_1(2^w - 1) \iff \forall b \in \mathcal{B}_1,\ b = 0. \tag{4.1}$$

Analogamente, o valor mínimo é definido como

$$c_{\min} = \sum_{i=t-t_1}^{t-1} (2^w - 1 - 2^w - 1) = 0 \iff \forall b \in \mathcal{B}_1,\ b = 2^w - 1. \tag{4.2}$$

A partir de $c_{\max}$, o número de *bits* necessários para representar qualquer valor

w	m	n_c	n_u	t_2w	w	m	n_c	n_u	t_2w	w	m	n_c	n_u	t_2w
2	128	8	2	10	7	128	12	2	14	12	128	16	8	24
	192	9	1			192	12	2			192	16	8	
	256	9	1			256	13	1			256	17	7	
	512	10	2	12		512	14	0			512	18	6	
3	128	9	0	9	8	128	12	4	16	13	128	17	9	26
	192	9	3			192	13	3			192	17	9	
	256	10	2	12		256	13	3			256	18	8	
	512	11	1			512	14	2			512	19	7	
4	128	9	3	12	9	128	13	5	18	14	128	18	10	28
	192	10	2			192	14	4			192	18	10	
	256	10	2			256	14	4			256	19	9	
	512	11	1			512	15	3			512	20	8	
5	128	10	0	10	10	128	14	6	20	15	128	19	11	30
	192	11	4			192	15	5			192	19	11	
	256	11	4	15		256	15	5			256	20	10	
	512	12	3			512	16	4			512	21	9	
6	128	11	1	12	11	128	15	7	22	16	128	19	13	32
	192	11	1			192	16	6			192	20	12	
	256	12	0			256	16	6			256	20	12	
	512	13	5	18		512	17	5			512	21	11	

Tabela 1 – *Bits* inutilizados em $\mathcal{B}_2$ para vários valores de w e m.

de c é calculado. Defina este valor como $n_c = \lceil \log_2 c_{\max} \rceil = \lceil \log_2(t_1 \times (2^w - 1)) \rceil$. Note que o número de *bits* reservados para este valor na definição do esquema é diretamente relacionado com o valor de t_2, *i.e.* t_2w. Observe que, de acordo com as definições de t_1 e t_2, sempre é verdade que $n_c \leq t_2w$:

$$\begin{aligned}
n_c &\leq t_2w \\
\lceil \log_2(t_1 \times (2^w - 1)) \rceil &\leq t_2w \\
\lceil \log_2 t_1 + \log_2(2^w - 1) \rceil &\leq t_2w \\
\lceil \log_2 t_1 + w \rceil &\leq t_2w \\
\lceil \log_2 t_1 \rceil + w &\leq t_2w \\
\lceil \log_2 t_1 \rceil + w &\leq \lceil \lfloor (\log_2 t_1 \rfloor + 1 + w) / w \rceil \times w \\
\lceil \log_2 t_1 \rceil + w &\leq \lceil \lfloor \log_2 t_1 \rfloor + 1 + w \rceil \\
\lceil \log_2 t_1 \rceil + w &\leq \lfloor \log_2 t_1 \rfloor + 1 + w \\
\lceil \log_2 t_1 \rceil + w &\leq \lceil \log_2(t_1 + 1) \rceil + w.
\end{aligned} \tag{4.3}$$

Isto ocorre pois, para acomodar c, um número inteiro de palavras em base-w deve ser utilizado. Portanto, a quantidade de *bits* inutilizados é definida como o número natural $n_u = t_2w - n_c$. Para os valores mais comuns de m e praticáveis para w, a Tabela 1 apresenta várias combinações de parâmetros para esquemas da família Winternitz.

4.2 WOTS-R

A segunda proposta consiste na busca por resumos criptográficos com características que habilitam a geração ou verificação de assinaturas de maneira mais eficiente. Este processo é independente do esquema subjacente, consistindo do pré-processamento da mensagem através de sua concatenação com uma palavra arbitrária. Deste modo, o valor do resumo criptográfico resultante pode ser maximizado ou minimizado, de acordo com a compensação desejada, consequentemente alterando a extensão das cadeias de resumos derivadas do mesmo. Esta estratégia é baseada no esquema publicado em (STEINWANDT; VILLáNYI, 2008).

Tome M como uma mensagem qualquer e o parâmetro de busca $R \in \mathbb{N}^*$. Na etapa de geração da assinatura, compute todos os resumos

$$D = \{\mathcal{H}(M||r) : 0 \leq r < R\} \tag{4.4}$$

e interprete-os como t_1 palavras de base-w, formando tuplas $\mathcal{B}_1^r$, como explicado na Subseção 3.1.2. A fim de escolher o elemento mais conveniente para que a geração de assinaturas seja eficiente, tome r de modo a produzir a menor soma entre as tuplas, representado por

$$k = \underset{r\in\{0,\ldots,R-1\}}{\operatorname{argmin}} \sum \mathcal{B}_1^r. \tag{4.5}$$

De modo análogo, para melhorar a verificação de assinaturas, obtenha r que gere a maior soma entre as tuplas, ou seja,

$$k = \underset{r\in\{0,\ldots,R-1\}}{\operatorname{argmax}} \sum \mathcal{B}_1^r. \tag{4.6}$$

Então, $d = \mathcal{H}(M||k)$ é considerado para a execução do esquema. Note que a concatenação de valores pseudoaleatórios pode ser realizada, entretanto aplicações de funções de resumo criptográfico tornam a saída completamente difusa, e portanto, a relevância do conteúdo das palavras é diminuída.

Observe que o cálculo repetido do conjunto de resumos D implica em um custo computacional ampliado na geração da assinatura. Portanto, é desejável que exista um limite para este parâmetro, visto que para alguns parâmetros w, este cálculo pode representar uma parte considerável do processamento total nesta etapa. Uma abordagem estatística é aplicada para este fim. Considerando que uma função de resumo criptográfico deve produzir saídas uniformemente distribuídas e aparentemente aleatórias, então frações destas saídas também devem o ser.

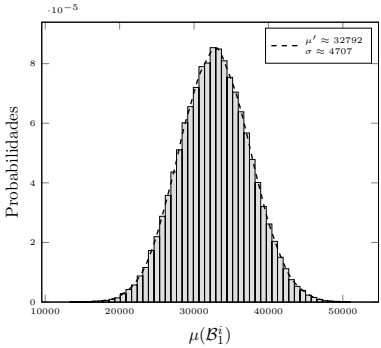

Figura 12 – Histograma normalizado de $\mu(\mathcal{B}_1^i)$, com 50 classes.

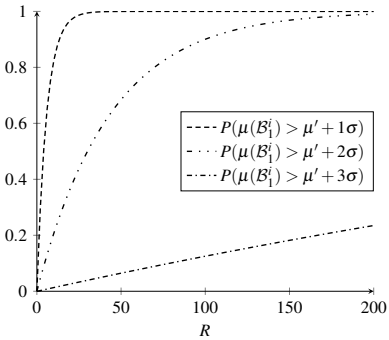

Figura 13 – Valores de R para que modificações de $\mathcal{B}_1^i$ sejam efetivas.

Portanto, se os valores em $\mathcal{B}_1$ seguem a distribuição uniforme discreta, pelo teorema central do limite, computações repetidas de $\mu(\mathcal{B}_1)$ tomarão a forma de uma distribuição Gaussiana. Então,

$$\mu' = \lim_{n\to\infty} \mu(\{\mu(\mathcal{B}_1^i) : 0 \leq i \leq n\}) = 2^{w-1}. \tag{4.7}$$

Para assegurar este comportamento, uma instância de WOTS foi produzida com os parâmetros $w = 16$, $m = 256$ e $\mathcal{H} =$ SHA-256, e os resumos

$$\{\mathcal{H}(\lambda_i) : 0 \leq i \leq 2^{16} - 1, \lambda_i \xleftarrow{\$} \{0,1\}^{32}\} \tag{4.8}$$

foram calculados. As médias das tuplas $\mathcal{B}_1^i$ são apresentadas na Figura 12, concretizando os fatos acima.

Assim, este resultado é utilizado a fim de fornecer valores razoáveis para R. Aplicando a tabela da distribuição normal padrão com desvio padrão σ, identificam-se as probabilidades de $\mu(\mathcal{B}_1^i)$ estar localizada após μ', e em especial, após intervalos de desvio padrão:

$$\begin{aligned} P(\mu(\mathcal{B}_1^i) > \mu' + 1\sigma) &= 0.1587, \\ P(\mu(\mathcal{B}_1^i) > \mu' + 2\sigma) &= 0.0228, \\ P(\mu(\mathcal{B}_1^i) > \mu' + 3\sigma) &= 0.0013. \end{aligned} \tag{4.9}$$

Então, para garantir que estes eventos aconteçam com probabilidade $\geq 98\%$, a distribuição binomial é utilizada a fim de procurar a quantidade mínima de experimentos em que pelo menos um destes produza o valor desejado.

Este comportamento pode ser visualizado na Figura 13, com quantidades sugeridas de $\{25, 200, 3500\}$ para, respectivamente, parâmetros $w = \{4, 8, 16\}$, recomendados em (HüLSING et al., 2018, Capítulo 6). Note que os múltiplos de σ utilizados facilitam o cálculo das probabilidades denotadas acima, contudo estes valores podem ser substituídos por outros que sejam adequados, a fim de construir novos critérios para a escolha de R.

Como a modificação não afeta mecanismos internos do esquema, é necessário discutir as consequências da concatenação de um dado λ_k à mensagem original, e do processo de busca por este dado. A maximização ou minimização das tuplas $\mathcal{B}_1^i$ introduz um comportamento estável nos *bits* mais significativos de cada elemento da tupla, *i.e.* torna-se mais comum que estes possuam valores antecipáveis. Isto é possivelmente explorável através de criptoanálise diferencial aplicada sobre *bits* fixos na construção Merkle—Damgård, posta em prática recentemente através da primeira colisão de resumos criptográficos produzidos pelo algoritmo SHA-1 (STEVENS et al., 2017). Portanto, considerar funções seguras na escolha de $\mathcal{H}$ é recomendável, como demonstrado em (HüLSING, 2013, Seção 3.2).

4.3 RESULTADOS

Para demonstrar a eficiência teórica dos esquemas propostos, o número de iterações da função interna f é comparado entre WOTS, WOTS-B, WOTS-R e o esquema WOTS-BR, onde as duas otimizações são aplicadas ao mesmo tempo. Parâmetros são escolhidos de acordo com a Seção 4.2. Considere mensagens aleatórias de 2^{10} caracteres, geradas através de `/dev/urandom`.

A Figura 14 considera o ganho obtido sobre a média aritmética de 2^{14} conjuntos $\mathcal{B}$, enfatizando o efeito de cada esquema variante. Note que $\mu(\mathcal{B}_2)$ é afetada negativamente por WOTS-R no caso da maximização dos elementos em $\mathcal{B}_1$. Entretanto, esta diferença não impacta os ganhos totais de maneira abundante, e a utilização de WOTS-BR mitiga esta perda. No caso da minimização de $\mu(\mathcal{B}_1)$, a otimização da soma de verificação não é aplicada, visto que atuará de maneira contraditória em relação à proposta.

Considerando a maximização de $\mu(\mathcal{B}_1)$, uma redução de aproximadamente 2^w iterações de f para o esquema WOTS-B é observada, condizente com o número de *bits* inutilizados apresentados na Tabela 1. No caso de WOTS-R, são observadas reduções de até 25% para $w = 4$, 33% para $w = 8$ e 42% para $w = 16$. Aplicando ambas as otimizações, o esquema WOTS-BR resulta em melhorias de até 28%, 39% e 52%, respectivamente. No caso da minimização, ocorrem refinamentos de 28%, 35% e 47%.

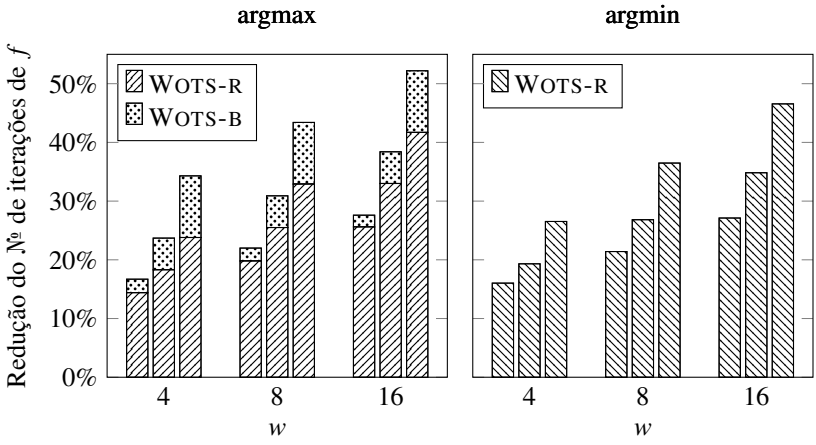

Figura 14 – Efeitos da aplicação das otimizações propostas no WOTS.

Note que a redução no cômputo destas iterações é diretamente traduzida na etapa complementar, ou seja, na maximização de $\mu(\mathcal{B}_1)$, existirá um aumento do número de iterações na geração da assinatura, proporcional à otimização obtida no passo de verificação. O mesmo conceito é aplicado para a minimização. Ademais, é preciso considerar a busca de k junto aos custos da geração da assinatura. Em vista disso, valores de R muito altos em relação ao parâmetro w são indesejáveis quando este é reduzido, *e.g.* com $w = 4$, o valor $R = 200$ torna-se uma fração substancial do custo total para a geração da assinatura. Ainda assim, casos de uso onde mostra-se necessário otimizar completamente o passo de verificação podem fazer uso de valores excedentes para R.

4.3.1 Impacto em esquemas baseados em árvores de Merkle

Evidentemente, as otimizações propostas nos esquemas WOTS-B e WOTS-R podem ser aplicadas a quaisquer variantes da família Winternitz. Consequentemente, qualquer esquema baseado em árvores de Merkle pode fazer uso de instâncias otimizadas destes em suas folhas. A fim de demonstrar esta afirmação, a implementação de WOTS+, esquema de assinatura única escolhido para a variante de XMSS, definida em (HüLSING et al., 2018, Seção 3.1) e explorada na Subseção 3.1.3, é modificada de acordo, e pode ser encontrada em `https://github.com/zambonin/xmss-reference`.

	w	R	ESQUEMA	TEMPO PARA $\mathcal{S}$	TEMPO PARA $\mathcal{V}$
argmax	4	—	XMSS	0.953	0.734
			XMSS-B	0.975	0.724
		25	XMSS-R	1.059	0.652
			XMSS-BR	1.073	0.637
		200	XMSS-R	1.222	0.620
			XMSS-BR	1.240	0.616
		3500	XMSS-R	3.724	0.588
			XMSS-BR	3.730	0.573
	8	—	XMSS	7.709	5.676
			XMSS-B	7.908	5.361
		25	XMSS-R	8.597	4.637
			XMSS-BR	8.992	4.415
		200	XMSS-R	9.045	4.245
			XMSS-BR	9.460	4.052
		3500	XMSS-R	11.760	3.861
			XMSS-BR	12.193	3.664
argmin	4	—	XMSS	0.971	0.746
		25		0.879	0.832
		200	XMSS-R	1.006	0.885
		3500		3.393	0.898
	8	—	XMSS	7.819	5.731
		25		6.672	6.553
		200	XMSS-R	6.472	6.982
		3500		8.488	7.435

Tabela 2 – Tempo de execução para otimizações incluídas no XMSS.

A Tabela 2 apresenta os resultados obtidos, na forma da média do tempo de execução em milissegundos de 2^{14} instâncias dos esquemas XMSS, XMSS-B, XMSS-R e XMSS-BR, de acordo com a notação fixada nas Seções 4.1 e 4.2. Note que $w = 16$ não é considerado visto que, para este parâmetro, a etapa de geração de chaves para o esquema XMSS é demasiado prolongada. Novamente, a otimização da soma de verificação não é considerada quando é desejável otimizar a geração de assinaturas.

Os experimentos foram realizados utilizando um computador com as seguintes especificações: 8 GB de RAM DDR3 @ 1333MHz, Intel Core i5 4570 @ 3.2GHz e `gcc` 7.3.0. Neste ambiente, observe que qualquer valor de R melhora o tempo de verificação ao aplicar a maximização de $\mu(\mathcal{B}_1)$ e otimização da soma de verificação, obtendo melhoras de até 22% com $w = 4$ e 36% para $w = 8$. Entretanto, a escolha de valores muito altos para R na aplicação da minimização de $\mu(\mathcal{B}_1)$ torna o passo de geração de assinatura mais lento. Assim, são recomendados os valores $R \leq 25$ para $w = 4$, e $R \leq 200$ para $w = 8$.

5 CONCLUSÃO

Neste trabalho, esquemas de assinatura digital baseados em funções de resumo criptográfico foram discutidos, a fim de prover um embasamento teórico para o desenvolvimento de otimizações sobre estes esquemas. Vários conceitos relevantes para a área são apresentados, como a estrutura de dados chamada de árvore de Merkle, que habilita a criação de esquemas de assinatura digital convencionais com múltiplas instâncias de assinaturas únicas, o encadeamento de árvores a fim de aumentar o número de mensagens passíveis de assinatura, e o gerenciamento sobre estas árvores quando seu armazenamento total se torna impraticável.

O resultado principal, consistindo na otimização da família de esquemas de assinatura digital única Winternitz, introduz um parâmetro de compensação que habilita o deslocamento do cômputo de iterações de f da geração da assinatura para a verificação, e vice-versa. Para fundamentar este trabalho, experimentos foram realizados em esquemas do estado da arte, com resultados positivos.

5.1 TRABALHOS FUTUROS

A otimização apresentada na Seção 4.2 pode considerar a modificação da tupla $\mathcal{B}_1$ de acordo com outros critérios, adicionais à maximização ou minimização de seus elementos, ou substituindo este processo completamente. Por exemplo, visto que o cômputo de iterações de f é feito sequencialmente, a busca de um critério a fim de obter elementos $b \in \mathcal{B}_1$ próximos em magnitude (*e.g.* a redução do desvio padrão) habilita o processamento paralelo destas iterações e, portanto, um tempo de execução reduzido para a geração e verificação de assinaturas. Ademais, este mesmo recurso pode ser utilizado para diminuir o custo da busca por $\mathcal{B}_1$ considerando o parâmetro R, também calculando estes resumos de maneira paralela.

Experimentos que exploram as propostas do Capítulo 4 podem ser realizados em variantes diferentes da família Winternitz, como WOTS-PR e WOTS-T, bem como em outros esquemas baseados em árvores de Merkle. Ademais, a segurança do esquema WOTS-R deve ser formalizada, visto que a modificação de $\mathcal{B}_1$ pode introduzir imperfeições nas cadeias de resumos subsequentes.

REFERÊNCIAS

BELLARE, M.; ROGAWAY, P. Optimal asymmetric encryption. In: DESANTIS, A. (Ed.). **Advances in Cryptology – EUROCRYPT '94**. [S.l.: s.n.], 1994. (Lecture Notes in Computer Science, v. 950), p. 92–111.

BERNSTEIN, D. J.; BUCHMANN, J.; DAHMEN, E. **Post Quantum Cryptography**. 1st. ed. [S.l.: s.n.], 2008. ISBN 3540887016.

BERNSTEIN, D. J. et al. **SPHINCS$^+$ – Submission to the NIST post-quantum project**. dez. 2017. Disponível em: <https://sphincs.org/data/sphincs+-specification.pdf>.

BERNSTEIN, D. J. et al. SPHINCS: Practical Stateless Hash-Based Signatures. In: OSWALD, E.; FISCHLIN, M. (Ed.). **Advances in Cryptology – EUROCRYPT 2015**. [s.n.], 2015. (Lecture Notes in Computer Science, v. 9056), p. 368–397. Disponível em: <https://eprint.iacr.org/2014/795>.

BERNSTEIN, D. J.; LANGE, T. Post-quantum cryptography. **Nature**, v. 549, n. 7671, p. 188–194, set. 2017.

BERTONI, G. et al. **Cryptographic sponge functions**. jan. 2011. Disponível em: <http://sponge.noekeon.org/>.

BERTONI, G. et al. **The KECCAK reference**. jan. 2011. Disponível em: <http://keccak.noekeon.org/>.

BLEICHENBACHER, D.; MAURER, U. M. Directed Acyclic Graphs, One-way Functions and Digital Signatures. In: DESMEDT, Y. G. (Ed.). **Advances in Cryptology – CRYPTO '94**. [S.l.: s.n.], 1994. (Lecture Notes in Computer Science, v. 839), p. 75–82.

BONEH, D. Twenty Years of Attacks on the RSA Cryptosystem. **Notices of the AMS**, v. 46, n. 2, p. 203–213, fev. 1999. Disponível em: <http://www.ams.org/notices/199902/boneh.pdf>.

BUCHMANN, J.; DAHMEN, E.; HüLSING, A. XMSS - a Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In: YANG, B.-Y. (Ed.). **Post-Quantum Cryptography**. [s.n.], 2011. (Lecture Notes in Computer Science, v. 7071), p. 117–129. Disponível em: <https://eprint.iacr.org/2011/484>.

BUCHMANN, J. et al. Merkle Signatures with Virtually Unlimited Signature Capacity. In: KATZ, J.; YUNG, M. (Ed.). **Applied Cryptography and Network Security**. [S.l.: s.n.], 2007. (Lecture Notes in Computer Science, v. 4521), p. 31–45.

BUCHMANN, J.; DAHMEN, E.; SCHNEIDER, M. Merkle Tree Traversal Revisited. In: BUCHMANN, J.; DING, J. (Ed.). **Post-Quantum Cryptography**. [S.l.: s.n.], 2008. (Lecture Notes in Computer Science, v. 5299), p. 63–78.

BUCHMANN, J. et al. CMSS: An Improved Merkle Signature Scheme. In: BARUA, R.; LANGE, T. (Ed.). **Progress in Cryptology – INDOCRYPT 2006**. [s.n.], 2006. (Lecture Notes in Computer Science, v. 4329), p. 349–363. Disponível em: <https://eprint.iacr.org/2006/320>.

CRUZ, J. P.; YATANI, Y.; KAJI, Y. Constant-sum fingerprinting for Winternitz one-time signature. In: **2016 International Symposium on Information Theory and Its Applications (ISITA)**. [S.l.: s.n.], 2016. p. 703–707.

DAEMEN, J.; RIJMEN, V. **The Design of Rijndael**. 1st. ed. [S.l.: s.n.], 2002. ISBN 3540425802.

DAHMEN, E. et al. Digital Signatures Out of Second-Preimage Resistant Hash Functions. In: BUCHMANN, J.; DING, J. (Ed.). **Post-Quantum Cryptography**. [S.l.: s.n.], 2008. (Lecture Notes in Computer Science, v. 5299), p. 63–78.

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

DWORKIN, M. J. **SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions**. [S.l.], jul. 2015. Disponível em: <http://dx.doi.org/10.6028/NIST.FIPS.202>.

DWORKIN, M. J. et al. **Advanced Encryption Standard (AES)**. [S.l.], nov. 2001. Disponível em: <http://dx.doi.org/10.6028/NIST.FIPS.197>.

GOLDREICH, O. **Foundations of Cryptography: Volume 2, Basic Applications**. 1st. ed. [S.l.: s.n.], 2004. ISBN 0521830842.

GOLDWASSER, S.; MICALI, S. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In: **Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing**. [S.l.: s.n.], 1982. p. 365–377.

GROVER, L. K. A Fast Quantum Mechanical Algorithm for Database Search. In: **Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing**. [s.n.], 1996. p. 212–219. Disponível em: <https://arxiv.org/abs/quant-ph/9605043v3>.

HüLSING, A. W-OTS$^+$ – shorter signatures for hash-based signature schemes. In: YOUSSEF, A.; NITAJ, A.; HASSANIEN, A. E. (Ed.). **Progress in Cryptology – AFRICACRYPT 2013**. [s.n.], 2013. (Lecture Notes in Computer Science, v. 7918), p. 173–188. Disponível em: <https://eprint.iacr.org/2017/965>.

HüLSING, A.; BUSOLD, C.; BUCHMANN, J. Forward secure signatures on smart cards. In: KNUDSEN, L. R.; WU, H. (Ed.). **Selected Areas in Cryptography**. [S.l.: s.n.], 2012. (Lecture Notes in Computer Science, v. 7707), p. 66–80.

HüLSING, A. et al. **XMSS: Extended Hash-Based Signatures**. [S.l.], maio 2018. Disponível em: <https://tools.ietf.org/html/rfc8391>.

HüLSING, A.; RAUSCH, L.; BUCHMANN, J. Optimal Parameters for XMSS^{MT}. In: CUZZOCREA, A. et al. (Ed.). **Security Engineering and Intelligence Informatics**. [s.n.], 2013. (Lecture Notes in Computer Science, v. 8128), p. 194–208. Disponível em: <https://eprint.iacr.org/2017/966>.

HüLSING, A.; RIJNEVELD, J.; SONG, F. Mitigating Multi-target Attacks in Hash-Based Signatures. In: CHENG, C.-M. et al. (Ed.). **Public-Key Cryptography – PKC 2016**. [S.l.: s.n.], 2016. (Lecture Notes in Computer Science, v. 9614), p. 387–416.

JEAN, J. **TikZ for Cryptographers**. abr. 2016. Disponível em: <https://www.iacr.org/authors/tikz/>.

KATZ, J.; KOO, C.-Y. **On Constructing Universal One-Way Hash Functions from Arbitrary One-Way Functions**. set. 2005. Cryptology ePrint Archive, Report 2005/328. Disponível em: <https://eprint.iacr.org/2005/328>.

LAFRANCE, P. **Digital Signature Schemes Based on Hash Functions**. Dissertação (Mestrado) — University of Waterloo, abr. 2017. Disponível em: <https://uwspace.uwaterloo.ca/handle/10012/11679>.

LAMPORT, L. **Constructing digital signatures from a one-way function**. [S.l.], out. 1979. Disponível em: <https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/>.

LEIGHTON, F. T.; MICALI, S. **Large provably fast and secure digital signature schemes based on secure hash functions**. jul. 1995. US Patent 5,432,852. Disponível em: <https://patents.google.com/patent/US5432852A>.

MCGREW, D. et al. State Management for Hash-Based Signatures. In: **Security Standardisation Research**. [s.n.], 2016. p. 244–260. Disponível em: <https://eprint.iacr.org/2016/357>.

MCGREW, D. A.; CURCIO, M.; FLUHRER, S. **Hash-Based Signatures**. [S.l.], abr. 2018. Work in Progress. Disponível em: <https://datatracker.ietf.org/doc/html/draft-mcgrew-hash-sigs>.

MENEZES, A. J.; VANSTONE, S. A.; VAN OORSCHOT, P. C. **Handbook of Applied Cryptography**. 1st. ed. [S.l.: s.n.], 1996. ISBN 0849385237.

MERKLE, R. C. **Secrecy, Authentication, and Public Key Systems**. Tese (Doutorado) — Stanford University, jun. 1979. Disponível em: <http://www.merkle.com/papers/Thesis1979.pdf>.

MERKLE, R. C. A Certified Digital Signature. In: BRASSARD, G. (Ed.). **Advances in Cryptology – CRYPTO '89**. [S.l.: s.n.], 1989. (Lecture Notes in Computer Science, v. 435), p. 218–238.

PERIN, L. P. et al. Tuning the Winternitz Hash-Based Digital Signature Scheme. In: **2018 IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2018.

PERRIG, A. The BiBa One-time Signature and Broadcast Authentication Protocol. In: **Proceedings of the Eighth ACM Conference on Computer and Communications Security**. [S.l.: s.n.], 2001. p. 28–37.

REYZIN, L.; REYZIN, N. Better Than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In: BATTEN, L.; SEBERRY, J. (Ed.). **Information Security and Privacy**. [S.l.: s.n.], 2002. (Lecture Notes in Computer Science, v. 2384), p. 144–153.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. **Communications of the ACM**, v. 21, n. 2, p. 120–126, fev. 1978.

ROMPEL, J. One-way Functions Are Necessary and Sufficient for Secure Signatures. In: ORTIZ, H. (Ed.). **Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing**. [S.l.: s.n.], 1990. p. 387–394.

SHOR, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. **SIAM Journal on Computing**, v. 26, n. 5, p. 1484–1509, out. 1997. Disponível em: <https://arxiv.org/abs/quant-ph/9508027v2>.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 5th. ed. [S.l.: s.n.], 2010. ISBN 0136097049.

STEINWANDT, R.; VILLáNYI, V. I. A one-time signature using run-length encoding. **Information Processing Letters**, v. 108, n. 4, p. 179–185, out. 2008.

STEVENS, M. et al. The First Collision for Full SHA-1. In: KATZ, J.; SHACHAM, H. (Ed.). **Advances in Cryptology – CRYPTO 2017**. [s.n.], 2017. (Lecture Notes in Computer Science, v. 10401), p. 570–596. Disponível em: <https://eprint.iacr.org/2017/190>.

VON ZUR GATHEN, J. **CryptoSchool**. 1st. ed. [S.l.: s.n.], 2015. ISBN 3662484234.