

**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

Agradeço à minha mãe, Luciana, pela introdução antecipada às linguagens, o uso prudente destas, e o desejo pela docência; ao meu pai, Roberto, pelo incentivo à destreza numérica e tecnológica; e a ambos, naturalmente, pelo perfeccionismo, firmeza e ambição, valores essenciais no caminho para este trabalho.

Agradeço aos Professores Ricardo Custódio, Daniel Panario e Jean Martina, e ao Mestre Lucas Perin, pela orientação e/ou co-autoria deste trabalho, e por prover a infraestrutura necessária para que este trabalho fosse desenvolvido, na forma do Laboratório de Segurança em Computação.

Agradeço aos amigos que fazem ou fizeram parte desta jornada, sem os quais não seria possível chegar até aqui. Em especial, à Ana Letícia, pelo seu imutável suporte e companheirismo, e pela contagiante sinceridade; à Larissa, pela sua tamanha paciência, e sabedoria em momentos delicados; ao Douglas, pelos momentos de descontração e por prover a oportunidade de trabalhar em um ambiente proveitoso; aos caríssimos Dúnia, Emmanuel, Matheus e Ramna, pelas incontáveis horas de discussões proveitosas sobre quaisquer assuntos acadêmicos ou habituais; e à Poliana, pela sua confiança e entusiasmo, manifestados na forma de seu inexorável, imensurável e imperturbável amor.



“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 criptográficos 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. . . . .	36
Figura 2	Estrutura da construção esponja, onde $i, j \in \mathbb{N}^*$ . . . . .	38
Figura 3	Funcionamento típico de um esquema de assinatura digital. . .	40
Figura 4	Etapas de assinatura e verificação do WOTS, para $0 \leq i \leq t - 1$ . . . . .	46
Figura 5	Assinatura no HORS, para $0 \leq i \leq k - 1, 0 \leq j, l \leq t - 1$ . . . . .	50
Figura 6	Árvore de Merkle $T$ , com altura $H = 2$ . . . . .	51
Figura 7	Caminho de autenticação para a folha de índice $i = 3$ . . . . .	54
Figura 8	Caminho de autenticação para a folha $j = 3 \cdot 2^{h_0} + 5$ , em uma hiper-árvore com $d = 2, h_0 = h_1 = 3$ . . . . .	59
Figura 9	Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1979 até 1997. . .	64
Figura 10	Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1998 até 2018. . .	64
Figura 11	Histograma normalizado de $\mu(\mathcal{B}_1^i)$ , com 50 classes. . . . .	70
Figura 12	Valores de $R$ para que modificações de $\mathcal{B}_1^i$ sejam efetivas. . . . .	70
Figura 13	Efeitos da aplicação das otimizações propostas no WOTS. . . . .	72
Figura 14	Efeito das otimizações incluídas no XMSS. . . . .	73



## LISTA DE TABELAS

Tabela 1	<i>Bits</i> inutilizados em $\mathcal{B}_2$ para vários valores de $w$ e $m$ . . . . .	68
----------	----------------------------------------------------------------------------------------	----



## LISTA DE ABREVIATURAS E SIGLAS

ECDSA	<i>Elliptic Curve Digital Signature Algorithm</i> . . . . .	34
SHA	<i>Secure Hash Algorithm</i> . . . . .	37
RSA	<i>Rivest—Shamir—Adleman</i> . . . . .	41
LD-OTS	<i>Lamport—Diffie One-time Signature Scheme</i> . . . . .	43
WOTS	<i>Winternitz One-time Signature Scheme</i> . . . . .	43
MSS	<i>Merkle Signature Scheme</i> . . . . .	43
WOTS-T	<i>Winternitz One-time Signature Scheme with Tight security</i> . . .	46
IETF	<i>Internet Engineering Task Force</i> . . . . .	46
WOTS-LM	<i>Leighton—Micali—Winternitz One-time Signature Scheme</i> . .	46
AES-NI	<i>Advanced Encryption Standard New Instructions</i> . . . . .	48
HORS	<i>Hash to Obtain Random Subset</i> . . . . .	48
HORST	<i>Hash to Obtain Random Subset with Trees</i> . . . . .	50
SPR-MSS	<i>Second Preimage Resistant Merkle Signature Scheme</i> . . . . .	55
XMSS	<i>eXtended Merkle Signature Scheme</i> . . . . .	55
WOTS-PR	<i>Winternitz One-time Signature Scheme with Pseudorandom Functions</i> . . . . .	56
CMSS	<i>Coronado—Merkle Signature Scheme</i> . . . . .	58
GMSS	<i>Generalized Merkle Signature Scheme</i> . . . . .	58
XMSS-MT	<i>eXtended Merkle Signature Scheme with Multi-Trees</i> . . . . .	58
BDS	<i>Buchmann—Dahmen—Schneider</i> . . . . .	58
XMSS-T	<i>eXtended Merkle Signature Scheme with Tight security</i> . . . . .	63
SPHINCS	<i>Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures</i> . . . . .	65
AES	<i>Advanced Encryption Standard</i> . . . . .	85
NIST	<i>National Institute of Standards and Technology</i> . . . . .	85
DES	<i>Data Encryption Standard</i> . . . . .	85



## LISTA DE SÍMBOLOS

$ \omega $	Tamanho da palavra $\omega$ . . . . .	34
$\parallel$	Concatenação de palavras . . . . .	39
$\xleftarrow{s}$	Seleção aleatória . . . . .	44
$\Sigma^{(l,t)}$	Todas as $t$ -tuplas de palavras de tamanho $l$ construídas com $\Sigma$ . . .	44
$f^\alpha$	Aplicação de $f$ $\alpha$ vezes . . . . .	45
$\gg$	Muito maior que . . . . .	48
$\mathbb{F}_n$	Corpo de Galois de ordem $n$ . . . . .	85
$\llcirclearrowleft$	Deslocamento circular à esquerda de <i>bits</i> . . . . .	86





## LISTA DE ALGORITMOS

Algoritmo 1	Funcionamento de TREEHASH <sub>h</sub> .....	51
Algoritmo 2	TREEHASH para o cálculo de $T_{H,0}$ .....	52
Algoritmo 3	Travessia de árvores de Merkle BDS.....	61
Algoritmo 4	Codificação do AES.....	86
Algoritmo 5	Decodificação do AES.....	87



## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b> .....	29
1.1	OBJETIVOS .....	31
<b>1.1.1</b>	<b>Objetivo geral</b> .....	31
<b>1.1.2</b>	<b>Objetivos específicos</b> .....	31
1.2	TRABALHOS RELACIONADOS .....	31
1.3	CONTRIBUIÇÕES DESTE TRABALHO .....	32
<b>2</b>	<b>PRIMITIVAS CRIPTOGRÁFICAS</b> .....	33
2.1	CRIPTOGRAFIA SIMÉTRICA E ASSIMÉTRICA .....	33
2.2	FUNÇÕES DE SENTIDO ÚNICO .....	34
<b>2.2.1</b>	<b>Construção a partir de cifras de bloco</b> .....	37
<b>2.2.2</b>	<b>Construção esponja</b> .....	38
2.3	ESQUEMAS DE ASSINATURA DIGITAL .....	40
<b>2.3.1</b>	<b>O criptossistema RSA</b> .....	41
<b>3</b>	<b>ASSINATURA DIGITAL BASEADA EM FUNÇÕES DE RESUMO CRIPTOGRÁFICO</b> .....	43
3.1	ESQUEMAS DE ASSINATURA ÚNICA .....	43
<b>3.1.1</b>	<b>LD-OTs</b> .....	43
<b>3.1.2</b>	<b>WOTS</b> .....	45
<b>3.1.3</b>	<b>WOTS+</b> .....	47
<b>3.1.4</b>	<b>HORS</b> .....	48
3.2	ESQUEMAS BASEADOS EM ÁRVORES DE MERKLE ....	50
<b>3.2.1</b>	<b>MSS</b> .....	52
<b>3.2.2</b>	<b>XMSS</b> .....	55
<b>3.2.3</b>	<b>XMSS-MT</b> .....	58
3.2.3.1	Travessia de árvores de Merkle Bds .....	59
3.2.3.2	Descrição do esquema .....	62
3.3	VISÃO TEMPORAL .....	64
<b>4</b>	<b>OTIMIZAÇÃO DO ESQUEMA WINTERNITZ</b> .....	67
4.1	WOTS-B .....	67
4.2	WOTS-R .....	69
4.3	RESULTADOS .....	71
<b>4.3.1</b>	<b>Impacto em esquemas baseados em árvores de Merkle</b> ....	72
<b>5</b>	<b>CONCLUSÃO</b> .....	75
5.1	TRABALHOS FUTUROS .....	75
	<b>REFERÊNCIAS</b> .....	77
	<b>APÊNDICE A – DESCRIÇÃO DA CIFRA AES</b> .....	85
	<b>APÊNDICE B – ARTIGO DO TCC</b> .....	91



## 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, *e.g.* 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 de Lamport (LAMPOR, 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*, que pode ser consultado no Apêndice B. 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. Funções de sentido único, base teórica para funções de resumo criptográfico, são discutidas na Seção 2.2, e exemplos de construções teóricas por trás destas funções são apresentados nas Subseções 2.2.1 e 2.2.2. Por fim, a definição formal de um esquema de assinatura digital é apresentada na Seção 2.3, agregando estas noções e apresentando o criptossistema RSA como exemplo, na Subseção 2.3.1.

### 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 irreversível 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. Este processo é explorado na Subseção 2.2.1.

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  $f : A \rightarrow B$ , 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 matemáticas 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” através de um canal qualquer. Em especial, o algoritmo RSA é discutido como um simples exemplo de esquema de assinatura digital, na Subseção 2.3.1.

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).

## 2.2 FUNÇÕES DE SENTIDO ÚNICO

Funções cuja aplicação é viável, porém a inversão desta torna-se um problema difícil em se tratando de complexidade computacional, são chamadas de funções de sentido único. Apresentam várias utilidades no âmbito de segurança da informação e criptografia, em especial para o resumo de dados, ou seja, a redução de uma mensagem de tamanho arbitrário para uma palavra

pequena e identificável. Neste contexto, 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.

Estas funções podem possuir várias propriedades, apresentadas abaixo de acordo com (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Seção 9.2). Tome uma função  $\mathcal{H} : X \rightarrow Y$ . Comumente, os elementos de  $Y$  são chamados de resumos.

- (i)  $\mathcal{H}$  deve ser necessariamente *determinística*;
- (ii) O cálculo de todo resumo deve ser *computacionalmente fácil*<sup>1</sup>;
- (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 computacionalmente 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 sentido único que os itens (i), (ii) e (iv) sejam respeitados. Por fim, funções de sentido único Adicionalmente resistentes à segunda pré-imagem e/ou colisões são definidas como funções de resumo criptográfico, adequadas para utilização no contexto de segurança da informação. Finalmente, a relação da propriedade (vii) com estas funções é discutida na Subseção 2.2.1.

<sup>1</sup>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.

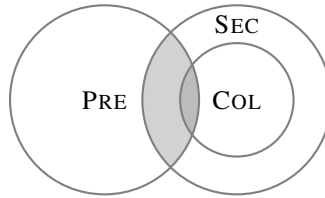


Figura 1 – Diagrama de Venn das resistências desejáveis para uma função de resumo no contexto de assinaturas digitais.

Funções de resumo criptográfico 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.

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).

Ademais, note que SEC e COL apresentam uma sutil diferença: na primeira, um adversário não pode escolher  $m_0$ , enquanto na segunda, quaisquer 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  $m_1$ .

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<sup>2</sup>; 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.

<sup>2</sup>Tabelas de resumos computados previamente, 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.

### 2.2.1 Construção a partir de cifras de bloco

Uma nova função de sentido único pode ter origem em algoritmos conhecidos como cifras de bloco, descritos como encadeamentos de operações matemáticas aplicadas sobre um estado de *bits* com tamanho fixo, geralmente chamado de bloco. Estes algoritmos comportam-se como uma família de permutações aleatórias parametrizados por uma chave, ou seja, o “embalramento” da mensagem em texto plano para um texto cifrado dependerá completamente da chave imposta sobre o algoritmo. Em especial, como visto em (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Exemplo 9.13), cifras com esta estrutura podem ser transformadas em funções de sentido único com a extensão da entrada limitado ao tamanho do bloco.

A extensão deste artifício para funções de sentido único com compressão pode ser realizada através de vários métodos. Destaca-se a construção Matyas-Meyer-Oseas (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Algoritmo 9.41), descrita a seguir. 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, divida  $x$  em uma  $k$ -tupla cujos elementos têm tamanho  $n$ , ou seja,  $X = (x_0, \dots, x_{k-1})$ . Preenchimento com zeros pode ser aplicado caso existam elementos que desrespeitem esta regra. O resultado deste processo é definido recursivamente como

$$\begin{aligned} h_0 &= IV, \\ h_i &= E_{g(h_{i-1})}(x_i) \oplus x_i, 1 \leq i \leq t. \end{aligned} \tag{2.1}$$

A partir deste alicerce, a construção Merkle—Damgård (MERKLE, 1979), base para as funções MD5, SHA1 e SHA2, pode ser aplicada a fim de construir uma função de resumo criptográfico convencional, através do encadeamento de aplicações de funções de sentido único com compressão sobre partes da mensagem de entrada. Este método deve também tratar a entrada original para que o tamanho desta seja um múltiplo do tamanho do bloco, com uma função de preenchimento, possivelmente contendo o tamanho da mensagem original como forma de fortalecimento deste arcabouço em se tratando de características de segurança.

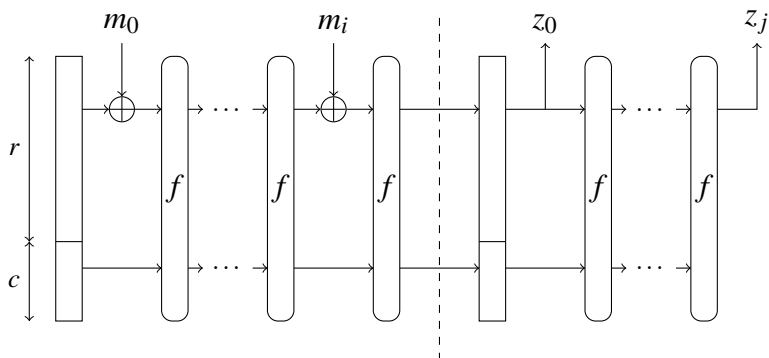


Figura 2 – Estrutura da construção esponja, onde  $i, j \in \mathbb{N}^*$ .

### 2.2.2 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.

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  $\text{SPONGE}[f, \text{pad}, r]$ , 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

$w = 2^\ell$ ,  $\ell \in \{0, \dots, 6\}$ . 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  $\theta$ .* Calcula o ou exclusivo entre um elemento de  $a$  e todos os elementos das colunas adjacentes a este.

*Etapa  $\rho$ .* Dispersa os elementos entre cortes transversais verticais de  $a$ .

*Etapa  $\pi$ .* Rearranja elementos em cortes transversais horizontais de  $a$ .

*Etapa  $\chi$ .* 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  $\iota$ .* Calcula o ou exclusivo entre o estado  $a$  e uma sequência gerada por um *linear-feedback shift register*<sup>3</sup> alimentado pelo índice da rodada atual, tornando a rodada assimétrica.

Tome  $\text{pad10*1}$  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

$$\text{KECCAK}[r, c](m, d) = \text{SPONGE}[\text{KECCAK} - f[r + c], \text{pad10*1}, r] \quad (2.2)$$

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

$$\text{KECCAK}[c] = \text{KECCAK}[1600 - c, c]. \quad (2.3)$$

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

$$\text{SHA3-256}(m) = \text{KECCAK}[512](m || 01, 256). \quad (2.4)$$

---

<sup>3</sup>Conjunto de registradores que deslocam *bits* a partir de funções lineares. Uma aplicação desta estrutura é a geração de números pseudoaleatórios.

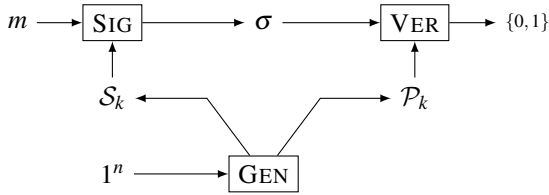


Figura 3 – Funcionamento típico de um esquema de assinatura digital.

### 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  $\text{GEN}(1^n)$ , 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  $\text{SIG}(\mathcal{S}_k, m)$ , que produz uma assinatura  $\sigma$  para uma mensagem  $m$ ; e o algoritmo de verificação  $\text{VER}(\mathcal{P}_k, m, \sigma)$ , 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,

$$\forall (p, s) \in \text{GEN}(1^n), \forall w \in \{0, 1\}^*, \quad \Pr[\text{VER}(p, w, \text{SIG}(s, w)) = 1] = 1. \quad (2.5)$$

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  $\sigma$  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.



### 2.3.1 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<sup>4</sup>. Em virtude de seu baixo desempenho computacional, 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,

$$\begin{aligned} \mathcal{S}_k &= d, \\ \mathcal{P}_k &= (n, e). \end{aligned} \tag{2.6}$$

*Geração da assinatura.* Para assinar uma mensagem, aplique uma função de resumo criptográfico  $\mathcal{H}$  à mensagem  $M$ , produzindo  $h = \mathcal{H}(M)$ . O texto cifrado  $c = h^d \pmod{n}$  é calculado e enviado para a entidade desejada na forma

$$\sigma = M || c. \tag{2.7}$$

*Verificação da assinatura.* O receptor da mensagem isola o resumo cifrado  $c$  de  $\sigma$ , e verifica a assinatura com sucesso se

$$h = c^e \pmod{n}. \tag{2.8}$$

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,  $m^{p-1} \equiv 1$

---

<sup>4</sup>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 à fatoração de inteiros (MENEZES; VANSTONE; VAN OORSCHOT, 1996, Fato 3.30).

(mod  $p$ ). Reescrevendo o produto  $ed$  como

$$ed = 1 + y\phi(n) = 1 + y(p-1)(q-1), y \in \mathbb{N}, \quad (2.9)$$

então

$$\begin{aligned} m^{ed} &\equiv m^{1+y(p-1)(q-1)} \equiv (m^{p-1})^{y(q-1)} m \\ &\equiv 1^{y(q-1)} m \equiv m \pmod{p}. \end{aligned} \quad (2.10)$$

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 uma descrição do algoritmo acima sem a presença de uma função de resumo criptográfico 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). Então, note que não existe qualquer fator aleatório na assinatura da mensagem, 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. A função  $\mathcal{H}$  previne este comportamento por conta de sua ampla difusão.

Ademais, este criptossistema é um exemplo adequado para a discussão entre codificação e assinatura de mensagens. Esta sutil diferença ocorre na utilização do par de chaves. A codificação de um dado com a chave pública impede que entidades externas à comunicação consigam ler o texto cifrado, entretanto, é impossível afirmar algo sobre a identidade do remetente da mensagem, ou o próprio conteúdo desta. Por outro lado, a assinatura de um dado com a chave privada implica que o signatário é detentor deste par de chaves, e isto pode ser verificado por qualquer indivíduo que possua a chave pública correspondente, livremente distribuída.

### 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, visto que vários esquemas continuam utilizando este recurso 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 (LAMPOR, 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  $S_k$ , 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  $S_k$  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<sup>1</sup> de sentido único  $f : \{0, 1\}^m \rightarrow \{0, 1\}^m$  e de resumo criptográfico  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^m$ . Seu funcionamento e algumas de suas características são discutidas abaixo.

*Geração de chaves.* Defina

$$\mathcal{S}_k = (x_{0,m-1}, x_{1,m-1}, \dots, x_{0,0}, x_{1,0}) \stackrel{\$}{\leftarrow} \{0, 1\}^{(m, 2m)} \quad (3.1)$$

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,

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

$$= (y_{0,m-1}, y_{1,m-1}, \dots, y_{0,0}, y_{1,0}). \quad (3.3)$$

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

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

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

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

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

<sup>1</sup>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 e seu funcionamento é representado na Figura 4.

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 sentido único  $f : \{0, 1\}^m \rightarrow \{0, 1\}^m$  e de resumo criptográfico  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^m$ .

*Geração de chaves.* Defina

$$\mathcal{S}_k = (x_{t-1}, \dots, x_0) \stackrel{\$}{\leftarrow} \{0, 1\}^{(m,t)} \quad (3.6)$$

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

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

*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- $w$ ,  $\mathcal{B}_1 = (b_{t-1}, \dots, b_{t-t_1})$ . 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

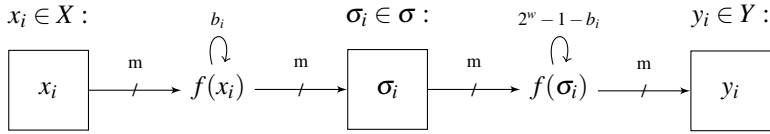


Figura 4 – Etapas de assinatura e verificação do WOTS, para  $0 \leq i \leq t-1$ .

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

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

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

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

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  $\sigma$  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  $\sigma'$  por uma entidade maliciosa é permitida, apenas através do cálculo de mais iterações de  $f$  em blocos da assinatura  $\sigma$ . Entretanto, com a presença de  $c$ , e conseqüentemente  $\mathcal{B}_2$ , 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 do esquema original (HÜLSING; RAUSCH; BUCHMANN, 2013), que utilizam apenas uma função. 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  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^m$  de maneira análoga ao esquema original. A família de funções que o WOTS+ utiliza é definida por

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

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

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

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

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

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

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

*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.

Aplicando a nova função de iteração, tem-se

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

*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,  $\sigma$  está correta se

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

Note que a utilização de uma função conhecida, como a cifra AES explicada no Apêndice A, para o cálculo de iterações de  $f$  é factível, conforme sugerido em (HÜLSING, 2013, Subseção 4.1). O método Matyas-Meyer-Oseas explicado na Subseção 2.2.1 é utilizado para que esta cifra seja adaptada, a fim de torná-la uma função da forma necessária pelo esquema WOTS+. Deste modo, implementações em hardware de rotinas relacionadas, como o conjunto de instruções AES-NI provido por processadores x86 recentes, podem ser aproveitados, resultando em um grande ganho de desempenho no cômputo de  $f$ .

### 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  $T = \{0, \dots, t-1\}$ , a



função de sentido único  $f : \{0, 1\}^m \rightarrow \{0, 1\}^m$  e a função de resumo criptográfico  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^m$ .

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  $(m_1, \dots, m_r)$ , encontrar  $r + 1$  mensagens tais que

$$\mathcal{H}(m_{r+1}) \subseteq \bigcup_{i=1}^r \mathcal{H}(m_i). \quad (3.14)$$

De outra forma, a probabilidade de que  $r$  assinaturas conttenham 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  $X = (x_{t-1}, \dots, x_0) \xleftarrow{\$} \{0, 1\}^{(m,t)}$ , e adicione  $k$  para obter a chave privada. Portanto,

$$\mathcal{S}_k = (k, x_{t-1}, \dots, x_0). \quad (3.15)$$

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

$$\mathcal{P}_k = (k, f(x_{t-1}), \dots, f(x_0)), \quad (3.16)$$

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

*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,  $\mathbf{d} = (d_{k-1}, \dots, d_0)$ . A assinatura é composta dos elementos de  $\mathcal{S}_k$  escolhidos pela representação inteira dos elementos de  $\mathbf{d}$ , ou seja,

$$\sigma = (x_{d_{k-1}}, \dots, x_{d_0}). \quad (3.17)$$

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

$$y_{d_i} = f(\sigma_i), \quad 0 \leq i \leq k - 1. \quad (3.18)$$

Parte deste processo é visualizado na Figura 5. 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, *e.g.*  $(d_0)_{10} = (d_{i-1})_{10} = l$ . Percebe-se, a partir da descrição dos algoritmos,

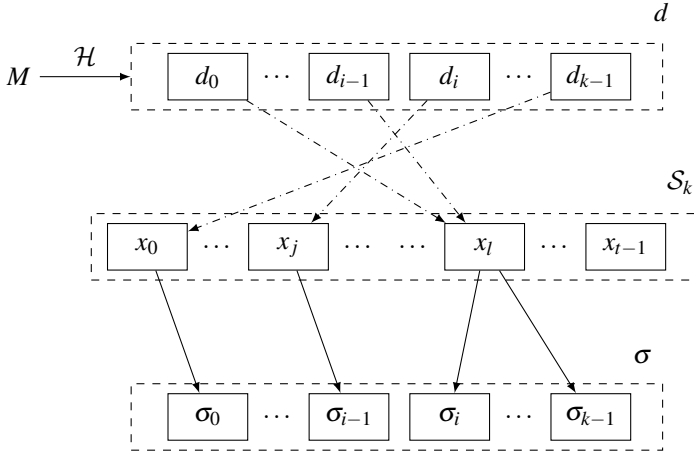


Figura 5 – Assinatura no HORST, para  $0 \leq i \leq k-1$ ,  $0 \leq j, l \leq t-1$ .

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 6. Considere uma árvore binária perfeita  $T$

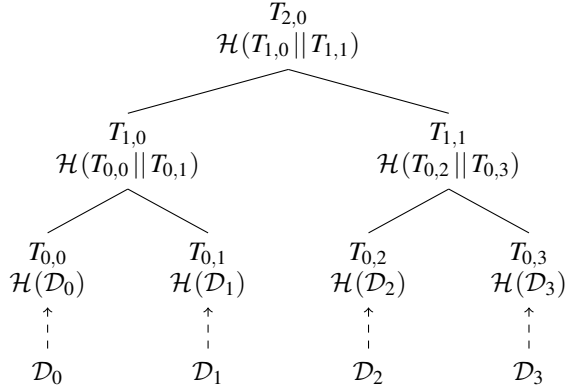


Figura 6 – Árvore de Merkle  $T$ , com altura  $H = 2$ .

de altura  $H \in \mathbb{N}^* \setminus \{1\}$ , onde um nodo é representado por

$$T_{h,j}, \quad 0 \leq h \leq H, \quad 0 \leq j \leq 2^{H-h} - 1, \quad (3.19)$$

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

$$\begin{aligned} T_{0,j} &= \mathcal{H}(\mathcal{D}_j), \text{ e} \\ T_{h,j} &= \mathcal{H}(T_{h-1,2j} || T_{h-1,2j+1}), \quad 1 \leq h \leq H. \end{aligned} \quad (3.20)$$

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 1** Funcionamento de TREEHASH<sub>h</sub>.

---

**Entrada:**  $h \in \{0, \dots, 2^H - 1\}$ , **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  $\text{LEAFCALC}(k)$ ,  $0 \leq k \leq 2^H - 1$ , que retorna o conteúdo da  $k$ -ésima folha, *i.e.*  $\mathcal{H}(\text{OTS}_{\mathcal{P}_k}^k)$ , e um nodo  $T_{\cdot}$  com a rotina  $\text{height}()$ , que retorna sua altura em  $T$ . O algoritmo  $\text{TREEHASH}_h$ , munido de uma pilha  $\mathbf{S}$  usual com as operações  $\mathbf{S.pop}()$ ,  $\mathbf{S.push}(\cdot)$  e  $\mathbf{S.top}()$  calcula um nodo de altura  $h$  em  $2^h$  chamadas de  $\text{LEAF}$  e  $2^h - 1$  cálculos de  $\mathcal{H}$ . Este processo pode ser visualizado no Algoritmo 1.

Em qualquer momento de sua execução, o número máximo de nodos armazenados em  $\mathbf{S}$  será  $h$ , chamados de nodos cauda, ou *tail nodes*. Ao seu estado final,  $\mathbf{S}$  conterá apenas o nodo desejado na altura  $h$ . Note que a execução de  $\text{TREEHASH}$  repetidamente, como visto no Algoritmo 2, 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 2**  $\text{TREEHASH}$  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 único genérico OTS, suas chaves privada e pública como  $\text{OTS}_{S_k}$  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  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^m$ .

*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

$$\mathcal{S}_k = ((\text{OTS}_{\mathcal{S}_k}^{2^H-1}, \text{OTS}_{\mathcal{P}_k}^{2^H-1}), \dots, (\text{OTS}_{\mathcal{S}_k}^0, \text{OTS}_{\mathcal{P}_k}^0)). \quad (3.21)$$

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,

$$\mathcal{P}_k = T_{H,0}. \quad (3.22)$$

*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  $\text{OTS}_{\mathcal{G}}^j$  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),

$$\text{AUTH}_h = \begin{cases} T_{h,j/2^h-1}, & \text{se } \lfloor s/2^h \rfloor \equiv 1 \pmod{2}, \\ T_{h,j/2^h+1}, & \text{se } \lfloor s/2^h \rfloor \equiv 0 \pmod{2}, \end{cases} \quad (3.23)$$

para  $0 \leq h \leq H-1$ , representado visualmente na Figura 7. A assinatura final leva em conta todos estes itens. Portanto,

$$\sigma = (j, \text{OTS}_{\mathcal{G}}^j, \text{OTS}_{\mathcal{P}_k}^j, \text{AUTH}). \quad (3.24)$$

*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:

$$\begin{aligned} a_0 &= \mathcal{H}(\text{OTS}_{\mathcal{P}_k}^j), \text{ e} \\ a_h &= \begin{cases} \mathcal{H}(a_{h-1} \parallel \text{AUTH}_{h-1}), & \text{se } \lfloor j/2^{h-1} \rfloor \equiv 0 \pmod{2}, \\ \mathcal{H}(\text{AUTH}_{h-1} \parallel a_{h-1}), & \text{se } \lfloor j/2^{h-1} \rfloor \equiv 1 \pmod{2}, \end{cases} \end{aligned} \quad (3.25)$$

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

$$\mathcal{P}_k = a_h, \quad (3.26)$$

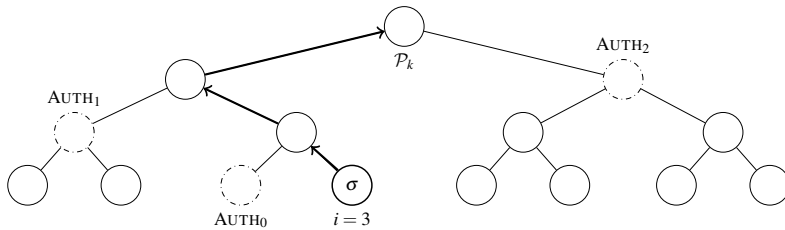


Figura 7 – Caminho de autenticação para a folha de índice  $i = 3$ .

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  $S_k$ , 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 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 esquema 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

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

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

$$\text{PRF}_a(b) = f_a(1) \parallel \dots \parallel f_a(b), \quad a \in \mathbb{N}, \quad b \in \{0, 1\}^m. \quad (3.27)$$

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  $r_i \xleftarrow{\$} \{0, 1\}^{2^m}$  para cada nível da árvore, ou seja,

$$T_{h,j} = h_k(r_i \oplus (T_{h-1,2j} \parallel T_{h-1,2j+1})), \quad 1 \leq h \leq H, \quad 0 \leq j \leq 2^{H-h} - 1. \quad (3.28)$$

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

$$\mathbf{r} = (r_0, \dots, r_{H+\lceil \log_2 t \rceil - 1}) \xleftarrow{\$} \{0, 1\}^{(2m, H+\lceil \log_2 t \rceil)}. \quad (3.29)$$

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

$$\text{OTS}_{\mathcal{S}_k}^i = \text{PRF}_t(f_s(i)), \forall i \in \{0, \dots, 2^H - 1\}. \quad (3.30)$$

A árvore  $T$  é construída utilizando a compressão de  $\text{OTS}_{\mathcal{P}_k}^i$  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,

$$\mathcal{S}_k = (s, 0), \quad (3.31)$$

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

$$\mathcal{P}_k = (T_{H,0}, \mathbf{r}). \quad (3.32)$$

*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,  $\text{OTS}_{\sigma}^j$  é calculada a partir de  $d$ . O caminho de autenticação AUTH é construído com o auxílio de algoritmos de travessia. Assim,

$$\sigma = (j, \text{OTS}_{\sigma}^j, \text{AUTH}). \quad (3.33)$$

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

$$\mathcal{P}_k = a_h, \quad (3.34)$$

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 arcações 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.

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.

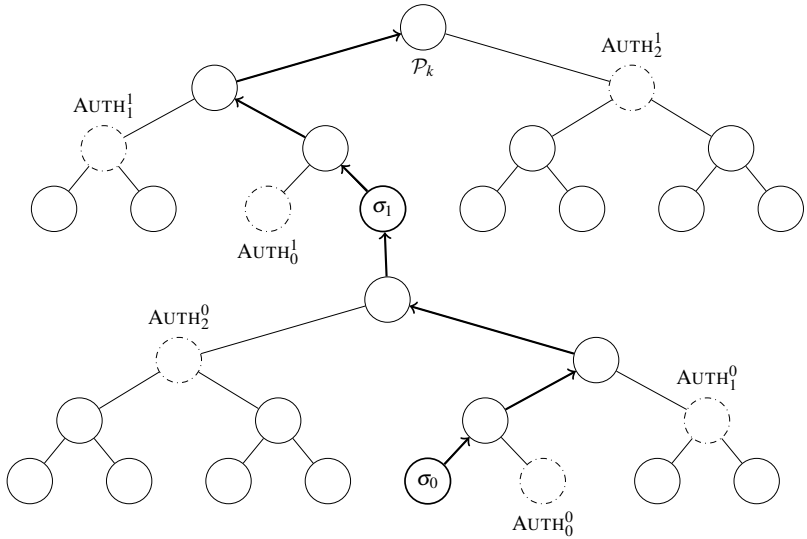


Figura 8 – Caminho de autenticação para a folha  $j = 3 \cdot 2^{h_0} + 5$ , em uma hiper-árvore com  $d = 2, h_0 = h_1 = 3$ .

### 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  $s \in \{0, \dots, 2^H - 2\}$ , e um parâmetro

$$K \in \mathbb{N}^* \setminus \{1\}, K < H, H - K/2 \equiv 0 \pmod{2} \quad (3.35)$$

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 STATE<sub>BDS</sub> 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)  $\mathbf{S}$ , pilha munida das operações usuais;
- (v)  $\{\text{TREEHASH}_h : 0 \leq h \leq H - 3\}$ , *i.e.* várias instâncias do Algoritmo 1.

As listas são munidas da operação  $\text{remove}(\cdot)$ , cujo argumento representa um índice válido de acordo com o tamanho da lista. As instâncias  $\text{TREEHASH}_h$  compartilham  $\mathbf{S}$ . São inicializadas por  $\text{init}(\cdot)$ , cujo argumento é um índice de folha que criará o nodo  $\text{LEAF}$  no Algoritmo 1, e podem ser executadas totalmente por  $\text{update}()$ . Ademais, são munidas da função  $\text{height}()$ , que retorna a menor altura entre nodos localizados nesta instância. Em casos anômalos, se  $\mathbf{S}$  estiver vazia,  $\text{TREEHASH}_h.\text{height}() = h$ , 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.  $\text{AUTH}$  recebe primeiramente o caminho de autenticação para a folha  $T_{0,0}$ , representado por  $\text{AUTH} = (T_{0,1}, \dots, T_{H-1,1})$ . As instâncias  $\text{TREEHASH}_h$  têm como nodos iniciais  $T_{h,3}$ . As pilhas  $\text{RETAIN}_h$  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 3, contempla ambos os casos, e será discutida em detalhes. Para uma folha  $T_{0,j}$ ,  $\tau$  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  $\tau$ . Caso este nodo esteja à esquerda, então  $\tau = 0$ , e do contrário,  $\tau = \max(\{2^h \mid j + 1 : 1 \leq h \leq H\})$ .

Para computar os nodos à esquerda, a utilização da lista  $\text{KEEP}$  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  $\text{AUTH}_\tau$  está à direita e pode ser armazenado em  $\text{KEEP}_\tau$ . Caso  $\tau = 0$ , então a folha está à esquerda e servirá como parte do caminho de autenticação para sua irmã. Portanto,  $\text{AUTH}_0$  recebe  $\text{LEAF}_{\text{CALC}}(s)$ . Do contrário, a folha está à direita, e o cálculo de  $\text{AUTH}_\tau$  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,  $\text{AUTH}_\tau$  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  $\text{RETAIN}_h$  e movidos para

---

**Algoritmo 3** Travessia de árvores de Merkle BDS.

**Entrada:**  $H \geq 2$ ,  $K$  (Eq. 3.35),  $s \in 0, \dots, 2^H - 2$ ,  $\text{STATE}_{\text{BDS}}$   $\triangleright$  altura da árvore  $T$ , parâmetro de compensação, índice de folha, estado do algoritmo

**Saída:**  $\text{AUTH}^{s+1}$   $\triangleright$  caminho de autenticação para a folha  $s + 1$

$\tau \leftarrow \max(\{2^h \mid s + 1 : h \in 1, \dots, H\})$

**se**  $\lfloor s/2^{\tau+1} \rfloor \equiv 0 \pmod{2}$  e  $\tau < H - 1$  **então**

$\text{KEEP}_{\tau} \leftarrow \text{AUTH}_{\tau}$

**fim se**

**se**  $\tau = 0$  **então**

$\text{AUTH}_0 \leftarrow \text{LEAFCALC}(s)$

**senão se**  $\tau > 0$  **então**

$\text{AUTH}_{\tau} \leftarrow \mathcal{H}(\text{AUTH}_{\tau-1} \parallel \text{KEEP}_{\tau-1})$

$\text{KEEP.remove}(\tau - 1)$

**para**  $h = 0$  até  $\tau - 1$  **faça**

**se**  $h < H - K$  **então**

$\text{AUTH}_h \leftarrow \text{TREEHASH}_h.\text{pop}()$

**senão**

$\text{AUTH}_h \leftarrow \text{RETAIN}_h.\text{pop}()$

**fim se**

**fim para**

**para**  $h = 0$  até  $\min(\tau - 1, H - K - 1)$  **faça**

**se**  $s + 1 + 3 \cdot 2^h < 2^H$  **então**

$\text{TREEHASH}_h.\text{init}(s + 1 + 3 \cdot 2^h)$

**fim se**

**fim para**

**fim se**

**para**  $i = 0$  até  $H - K/2$  **faça**

$k \leftarrow \min(\{\text{TREEHASH}_j.\text{height}() : j \in 0, \dots, H - K - 1\})$

$\text{TREEHASH}_k.\text{update}()$

**fim para**

---

$\text{AUTH}_h$ . Nós com altura  $h < H - K$  fazem uso dos nós armazenados nas instâncias  $\text{TREEHASH}_h$ , então subsequentemente inicializadas com os nós  $s + 1 + 3 \cdot 2^h$ , se necessário. Finalmente, algumas instâncias cujos nós têm menor altura são atualizadas, de acordo com o critério em (BERNSTEIN; BUCHMANN; DAHMEN, 2008, Seção 4.5), a fim de adicionar nós 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  $\mathcal{H} : \{0, 1\}^{2m} \rightarrow \{0, 1\}^m$ , 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

$$k = \max(\{h_i + \lceil \log_2 t_i \rceil : 0 \leq i \leq d - 1\}), \quad (3.36)$$

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:  $\text{STATE} = (\text{STATE}_{\text{BDS}}^0, \dots, \text{STATE}_{\text{BDS}}^{d-1})$ . Utilizando instâncias do algoritmo  $\text{TREEHASH}$  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  $\Sigma = (\sigma_1^T, \dots, \sigma_{d-1}^T)$ .

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  $\text{TREEHASH}$  são também armazenados em uma estrutura

$$\text{NEXT} = (s_0^n, \dots, s_{d-2}^n, \text{TREEHASH}_0^n, \dots, \text{TREEHASH}_{d-2}^n, \text{STATE}_{\text{BDS}}^{0,n}, \dots, \text{STATE}_{\text{BDS}}^{d-2,n}), \quad (3.37)$$

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

$$\mathcal{S}_k = (\mathbf{s}, \text{STATE}, \Sigma, \text{NEXT}), \quad (3.38)$$

$$\mathcal{P}_k = (T_{h_{d-1},0}, \mathbf{r}). \quad (3.39)$$

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

$$\sigma = (j, \text{OTS}_\sigma^j, \Sigma, \text{AUTH}^0, \dots, \text{AUTH}^{d-1}). \quad (3.40)$$

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  $\mathbf{s}'$ , STATE',  $\Sigma'$ , NEXT'. Mais detalhes sobre este processo podem ser encontrados em (HÜLSING; RAUSCH; BUCHMANN, 2013, Capítulo 2).

*Verificação da assinatura.* Para verificar  $\sigma$  perante à hiper-árvore,  $\text{OTS}_\sigma^j$  é aferida a fim de construir  $\text{OTS}_{\mathcal{P}_k}$ . Com este nodo e  $\text{AUTH}^0$ , a raiz da árvore é construída, a fim de verificar a primeira assinatura em  $\Sigma$ . Este processo é repetido  $d - 1$  vezes, utilizando todos os caminhos de autenticação e assinaturas em  $\Sigma$  fornecidos por  $\sigma$ , 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 8.

Adicionalmente, como mencionado na Subseção 3.1.2, o esquema 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.

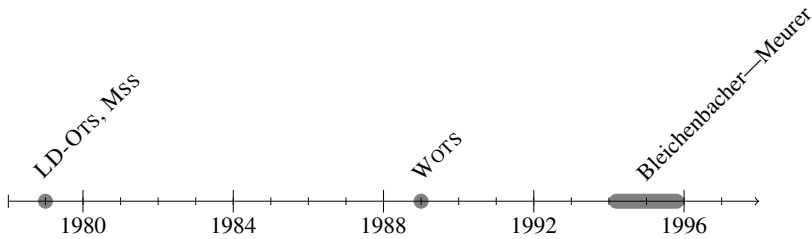


Figura 9 – Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1979 até 1997.

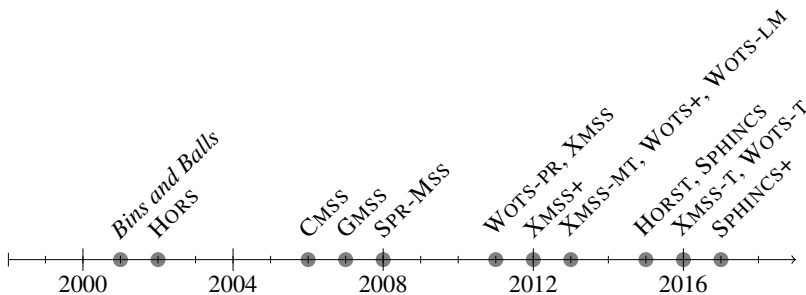


Figura 10 – Principais trabalhos acadêmicos sobre esquemas de assinatura digital baseados em funções de resumo criptográfico de 1998 até 2018.

### 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 9, nas duas décadas iniciais, esta área é estabelecida pela pesquisa em (LAMPOR, 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 9 e 10 contemplam todos os esquemas citados nas Seções 3.1 e 3.2, à exceção da 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

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

Analogamente, o valor mínimo é definido como

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

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			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	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	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	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  $n_c \leq t_2w$ :

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

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

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

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

$$k = \operatorname{argmin}_{r \in \{0, \dots, R-1\}} \sum \mathcal{B}_1^r. \quad (4.5)$$

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

$$k = \operatorname{argmax}_{r \in \{0, \dots, R-1\}} \sum \mathcal{B}_1^r. \quad (4.6)$$

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.

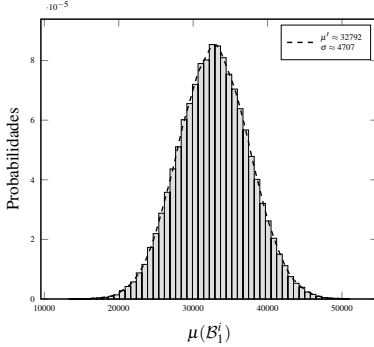


Figura 11 – Histograma normalizado de  $\mu(\mathcal{B}_1^i)$ , com 50 classes.

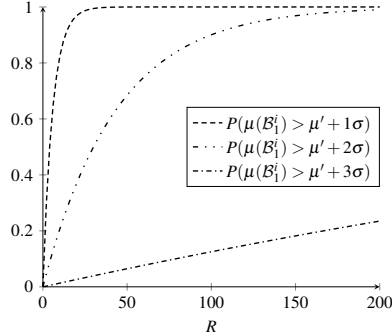


Figura 12 – Valores de  $R$  para que modificações de  $\mathcal{B}_1^i$  sejam efetivas.

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,

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

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

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

foram calculados. As médias das tuplas  $\mathcal{B}_1^i$  são apresentadas na Figura 11, 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  $\sigma$ , identificam-se as probabilidades de  $\mu(\mathcal{B}_1^i)$  estar localizada após  $\mu'$ , e em especial, após intervalos de desvio padrão:

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

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 12, 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  $\sigma$  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  $\lambda_k$  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 13 apresenta 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%.

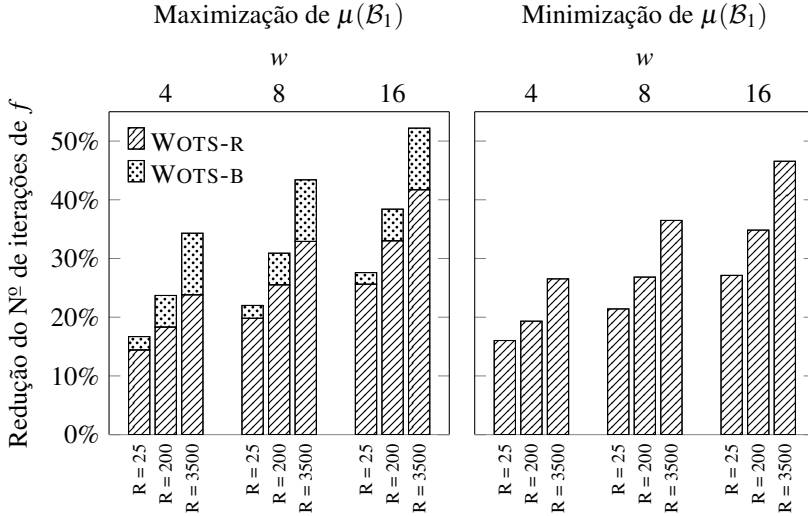


Figura 13 – Efeitos da aplicação das otimizações propostas no WOTS.

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, definido em (HÜLSING et al., 2018, Se-



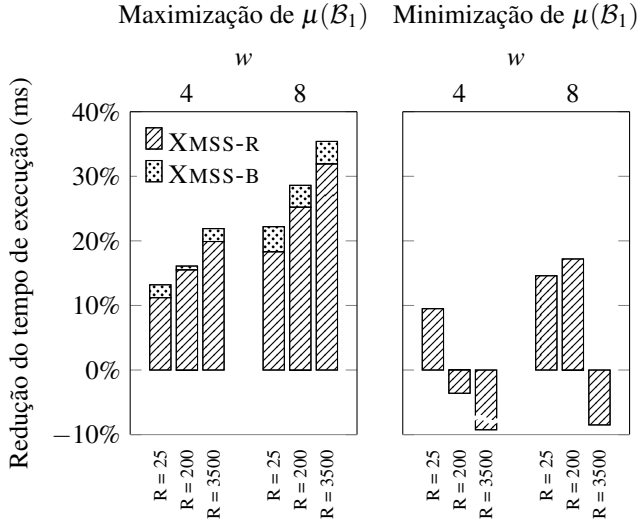


Figura 14 – Efeito das otimizações incluídas no XMSS.

ção 3.1) e explorado na Subseção 3.1.3, é modificada de acordo, e pode ser encontrada em <https://github.com/zambonin/xmss-reference>.

A Figura 14 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 8.1.1. 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. Em especial, a Figura 14 apresenta um valor fora da escala do gráfico, demonstrando uma grande perda neste aspecto. Assim, são recomendados os valores  $R \leq 25$  para  $w = 4$ , e  $R \leq 200$  para  $w = 8$ .

A compensação justa de processamento entre etapas complementares, preservada em esquemas da família Winternitz, apenas ocorre em esquemas

baseados em árvores de Merkle na minimização de  $\mathcal{B}_1$  com  $R$  de grandeza razoável. Esta impossibilidade do uso de valores excedentes para o parâmetro estabelece uma discussão sobre o impacto das estruturas de árvore sobre as otimizações propostas anteriormente. Isto ocorre devido à fatores necessários para o funcionamento prático dos esquemas, como possíveis implementações ingênuas do processo de busca do valor  $k$ , gerando considerável ônus no passo de criação de assinatura, conseqüentemente reduzindo o ganho obtido pelas propostas.

Além disso, note que a implementação de XMSS utilizada pode apresentar como comportamento opcional a execução do algoritmo de travessia discutido na Subseção 3.2.3.1, a fim de habilitar um ganho de desempenho geral ao longo das etapas do esquema. Testes iniciais com a implementação ingênuas demonstraram que os ganhos eram diretamente proporcionais. Assim, a travessia BDS é utilizada nos experimentos a fim de permitir testes mais extensivos, e demonstra a coexistência entre as otimizações propostas e este artifício.

## 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 demonstrar 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. 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.

Finalmente, vários aspectos são passíveis de discussão em se tratando da estrutura inerente de esquemas baseados em árvores de Merkle, como o estudo detalhado de parâmetros ótimos de  $R$  no caso da minimização de

$\mathcal{B}_1$ , o comportamento das otimizações apresentadas em esquemas com hiper-árvores, e o impacto detalhado destas sobre implementações que utilizem algoritmos de travessia distintos.

## REFERÊNCIAS

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<sup>+</sup> – 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.

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<sup>+</sup> – 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<sup>MT</sup>. 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.



## **APÊNDICE A – DESCRIÇÃO DA 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  $A_{i,j} \in \mathbb{F}_{2^8}^1$ ,  $0 \leq i, j \leq 3$ , 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. **SHIFTRROWS**: 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  $K_e = \{k_0, \dots, k_{t-1}\}$ . A lista de constantes  $RC$  com elementos em  $\mathbb{F}_{2^8}$  é definida pela recursão

$$RC_0 = x^0, \quad RC_1 = x^1, \quad RC_j = x \cdot RC_{j-1}, \quad j \geq 2. \quad (\text{A.1})$$

---

<sup>1</sup>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  $\ell$  palavras, que compõem o começo de  $K_e$ . Para os elementos restantes, ou seja,  $\forall i \geq \ell$ ,

$$k_i = k_{i-\ell} + \begin{cases} \text{SUBBYTES}(k_{i-1} \overset{\curvearrowright}{\ll} 8) + RC_i, & \text{se } i \equiv 0 \pmod{4}, \\ \text{SUBBYTES}(k_{i-1}), & \text{se } \ell = 8 \text{ e } i \equiv 4 \pmod{8}, \\ k_{i-1}, & \text{caso contrário.} \end{cases} \quad (\text{A.2})$$

Assim, uma função que criptografa uma mensagem  $m$  e retorna um texto cifrado  $c$  pode ser representada pelo Algoritmo 4. 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 4** 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

$A \leftarrow m$

$\{k_0 \dots k_{(n_r+1) \cdot \ell}\} \leftarrow \text{KEYEXPANSION}(K)$

$A \leftarrow \text{ADDROUNDKEY}(A, \{k_0, \dots, k_{\ell-1}\})$

**para**  $i \leftarrow 1$  até  $n_r - 1$  **faça**

$A \leftarrow \text{SUBBYTES}(A)$

$A \leftarrow \text{SHIFTRROWS}(A)$

$A \leftarrow \text{MIXCOLUMNS}(A)$

$A \leftarrow \text{ADDROUNDKEY}(A, \{k_{i \cdot \ell}, \dots, k_{(i+1) \cdot \ell - 1}\})$

**fim para**

$A \leftarrow \text{SUBBYTES}(A)$

$A \leftarrow \text{SHIFTRROWS}(A)$

$A \leftarrow \text{ADDROUNDKEY}(A, \{k_{n_r \cdot \ell}, \dots, k_{(n_r+1) \cdot \ell - 1}\})$

$c \leftarrow A$

---

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

- i. INVSHIFTRROWS: 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

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

Por fim, o resultado é representado pelo Algoritmo 5. 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 5** 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

```

 $A \leftarrow c$ 
 $\{k_0 \dots k_{(n_r+1) \cdot \ell}\} \leftarrow \text{KEYEXPANSION}(K)$ 
 $A \leftarrow \text{ADDRoundKEY}(A, \{k_{n_r \cdot \ell}, \dots, k_{(n_r+1) \cdot \ell - 1}\})$ 
para  $i \leftarrow n_r - 1$  até 1 faça
   $A \leftarrow \text{INVSHIFTROWS}(A)$ 
   $A \leftarrow \text{INVSUBBYTES}(A)$ 
   $A \leftarrow \text{ADDRoundKEY}(A, \{k_{i \cdot \ell}, \dots, k_{(i+1) \cdot \ell - 1}\})$ 
   $A \leftarrow \text{INVMIXCOLUMNS}(A)$ 
fim para
 $A \leftarrow \text{INVSHIFTROWS}(A)$ 
 $A \leftarrow \text{INVSUBBYTES}(A)$ 
 $A \leftarrow \text{ADDRoundKEY}(A, \{k_0, \dots, k_{\ell - 1}\})$ 
 $m \leftarrow A$ 

```

---

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).





## **APÊNDICE B – ARTIGO DO TCC**



# Tuning the Winternitz hash-based digital signature scheme

Lucas Pandolfo Perin, Gustavo Zambonin, Douglas Marcelino Beppler Martins,  
Ricardo Custódio, Jean Everson Martina  
Departamento de Informática e Estatística  
Universidade Federal de Santa Catarina  
Florianópolis, SC, 88040-900, Brasil  
E-mail: lucas.perin@posgrad.ufsc.br

**Abstract**—It is known that, for a given set of parameters, the overall complexity for generating and verifying a signature is constant and independent of the document being signed, for the Winternitz one-time signature scheme (WOTS). These costs are due to the number of chained iterations of a function  $f$ . However, the cost for signature generation alone is slightly different from signature verification, and these depend directly on the document. We introduce a new variant for WOTS, which allows the adjustment of these costs, i.e. increase the overall signature generation time in favor of faster verification or vice-versa. We decrease the number of iterations of  $f$  by up to half, with regards to the verification procedure, for commonly used values of the Winternitz parameter  $w$ . Additionally, our experiments show that these proposals have substantial impact on Merkle-based signature schemes, such as XMSS.

**Index Terms**—cryptography, digital signatures, public key

## I. INTRODUCTION

Traditional digital signature schemes, widely used in situations requiring secure and tamper-free communication (e.g. software distribution and SSL tunneling), are derived from cryptographic public key algorithms based on the computational hardness assumption of problems in number theory. Some examples are the factorization of large composite integers, closely related to the RSA cryptosystem, and the discrete logarithm, associated with the DSA and ECDSA algorithms. However, with the advent of quantum computers, these problems can be solved more efficiently with the aid of Shor's algorithm [1]. Ergo, such digital signature schemes will become fragile and may no longer guarantee the authenticity, integrity and non-repudiation properties required for digital signatures. The alternative is to use post-quantum cryptography — algorithms that are thought to be resistant against such techniques.

There are various methods for quantum-resilient digital signatures being proposed in the literature [2]. Some of the most popular ones make use of code-based cryptography, cryptographic hash functions, lattice structures and multivariate polynomials over finite fields. However, all these algorithms still need to be improved. According to [3], code-based cryptographic systems feature impractical key sizes, more security analysis on lattice-based signatures and multivariate cryptography are required, and hash-based signatures force the

signer to preserve a state of which keys were used, featuring large signature sizes if this behavior is undesired. Moreover, for the latter system, we believe that the cost difference between signature generation and verification is not exploited to tailor specific needs, such as in the case when signatures are repeatedly verified.

Notwithstanding these limitations, the robust flexibility and the abundant availability of cryptographic hash functions in all kinds of computational systems make hash-based signature schemes potential candidates to replace traditional digital signature schemes. For example, the inclusion of the SHA algorithm to the x86 SSE instruction set allows a possible improvement on the performance of these schemes with finetuned implementations and carefully chosen parameters. Additionally, it has been shown in the literature that one-way functions (the theoretical foundation of a hash function) are the only constructions needed to build a secure digital signature scheme [4], [5]. These reasons are strong indicators that hash-based signature schemes are presumably safe, and extremely fast in practice.

It is also important to note that organizations such as the U.S. National Institute of Standards and Technology (NIST) and the Internet Engineering Task Force (IETF) have shown great interest in the research and standardization of these algorithms. The SPHINCS+ proposal [6], a variant of [7] that reduces the signature size and execution time, was submitted to NIST as a candidate for this process. Besides, there are other examples of submissions [8] and Internet Drafts [9], [10] for hash-based schemes. These circumstances have motivated the proposition of several post-quantum digital signature schemes based on the use of hash functions [11], [2], [12].

The idea of using only hashes to sign digital messages is not new. Lamport [13] proposed, in 1979, a very simple and secure scheme to generate and verify digital signatures. It considers a pair of pseudorandom words as the private key and their hashes as the public key. The signer signs a single bit of information by choosing to distribute one string from the private key, on whether the bit is 0 or 1, as the signature. The verifier then computes the hash of this string, comparing it with part of the public key. By extending this idea to sign messages of arbitrary sizes, some drawbacks emerge, such as large sizes for the cryptographic key pair and signature.

These drawbacks have been the subject of research and several improvements have been proposed since its initial design. Perhaps the most promising method is the Winternitz one-time signature scheme (WOTS) due to Merkle [14]. This scheme is a generalization of the Lamport scheme. Unlike Lamport, WOTS allows more than one bit to be signed simultaneously. As such, the sizes of the cryptographic keys are decreased. In fact, the lower cost of the key has been replaced by a higher cost of processing. More recently, various hash-based signature schemes have been proposed based on, or using WOTS as part of the scheme [15], [16], [17], [18], [7], [6].

Most hash-based one-time signature schemes feature the following construction: the private key is composed of words generated by a pseudorandom function, and the public key is the result of recursively applying a hash function to the each element of the private key. To sign a document, a fingerprint function is used, which defines how many times the hash function should be applied to the private key to produce the signature. The result of this evaluation is the signature of the message. In order to perform the verification step, it is necessary to compute more hashes for the signature, until it coincides with the public key of the signer. Also, by applying a data structure known as Merkle tree, one can sign many documents with multiple one-time key pairs, and a single public key.

In the case of WOTS, the total number of hashes to sign a document and verify a signature depends solely on the size of the parameter  $w$ . However, these quantities are not distributed evenly, since they are derived from the result of the application of the fingerprint function to a message. Hence, these can be tailored to a faster signature generation or verification by carefully choosing a message derived from the original.

We propose two methods to improve the signature generation or verification steps for WOTS. Our first method changes the checksum computation slightly, by padding unused bits with ones in the place of zeros. This last proposal can be used to improve signature verification run times, while padding with zeros is already optimal for signature generation. Our main approach is based on [19], where we append a cryptographic nonce to a document to be signed and hash it. We repeat the process and search for hash outputs that can optimize signature generation or verification. In fact, by doing this, we do not change the underlying WOTS algorithm. We propose a fixed amount of operations before the signature is generated, so that the chosen step can be computed much faster. This is actually a trade-off choice, where improving one step results in more computations for the other. However, there is always an additional fixed cost to the signature generation. Both proposals can be applied independently or together for better results.

#### A. Related Works

Signatures are generated only once and frequently on a targeted device, which can be properly configured to accommodate their required resources. However, the resources available for the signature verification are unlikely to be

predictable in advance, since this step can be performed by any other kind of devices. For this reason, in practice, it is common to choose parameters that make it easier for a third party to verify signatures. An example is the use of the public exponent 65,537 of the RSA signature scheme. This choice makes signature verification faster than RSA signature generation. This is no different with hash-based signature schemes. Several papers proposed in the literature aim to improve the performance of the verification step for WOTS, such as [20], [21], [19]. While these works modify the nature of the Winternitz scheme, our main proposal consists of a pre-processing of the message, keeping the scheme intact. Another advantage of our scheme is the possibility of using larger values for the parameter  $w$  (such as  $w = 16$ ) substantially reducing the size of the WOTS signature and, even so, with small verification time at the expense of a slight increase in signature generation time.

#### B. Outline Remarks

*Notation.* We use the following symbols throughout the rest of this work. The length of a word  $\omega$  is given by  $|\omega|$ . For a set of integers  $S$ ,  $\min(S)$  and  $\max(S)$  are, respectively, the minimum and maximum elements of  $S$ . The concatenation of two words  $\omega_1, \omega_2$  is given by the symbol  $\|$ . Repeated mapping of a function  $f$  on some input  $x$  is recursively defined as  $f^0(x) = x$  and  $f^i(x) = f(f^{i-1}(x))$  for  $i \in \mathbb{N}^*$ . The symbol  $\xleftarrow{\$}$  is read as “chosen randomly from”. Finally, let  $\Delta^{(x_1, x_2)}$  be “a  $x_2$ -tuple of words with length  $x_1$  from alphabet  $\Delta$ ”.

*Organization.* The remaining of the paper is organized as follows: Section II briefly states the classical WOTS scheme; in Section III, we review some definitions of the scheme, present bounds for the variables used during the signature generation and verification steps and propose the modification of the checksum padding; Section IV contains the main proposal of this work; finally, Section V contains results obtained by experimenting with WOTS and XMSS, followed by our final considerations.

## II. CLASSICAL WINTERNITZ

In its original proposal [14], WOTS works as follows: take  $m$  as a security parameter, usually the output size in bits of a cryptographic hash function, and choose a parameter  $w \in \mathbb{N}^*$ ,  $w > 1$ . In addition, consider a non-compressing one-way function  $f : \{0, 1\}^m \rightarrow \{0, 1\}^m$  and a cryptographic hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^m$ . Then, compute

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

These parameters are selected so that  $w$  is the number of bits to be signed simultaneously. They are also directly related to the key size and signature performance, as we will see in the following.

*Key Generation.* Let  $X = (x_{t-1}, \dots, x_0) \xleftarrow{\$} \{0, 1\}^{(m,t)}$  be the private key. The public key  $Y = (y_{t-1}, \dots, y_0)$  is derived from  $X$  by applying  $f$  to each of the  $x_i$  elements  $2^w - 1$  times. Hence,  $Y = (f^{2^w-1}(x_{t-1}), \dots, f^{2^w-1}(x_0))$ .

*Signature Generation.* Take a message  $M$  and compute its digest  $d = \mathcal{H}(M)$ . For convenience, we choose  $m$  or  $w$  such that  $w \mid m$ . However, one may pad the digest by appending zeros to satisfy this requirement.  $d$  is then split into a  $t_1$ -tuple of base- $w$  words  $\mathcal{B}_1 = (b_{t_1-1}, \dots, b_{t_1-1})$ . Additionally, we compute the checksum using the integer representation of the elements in  $\mathcal{B}_1$ :

$$c = \sum_{i=t-t_1}^{t-1} 2^w - 1 - b_i.$$

Again,  $c$  may be padded with zeroes until  $w \mid |c|$ . Finally, it is split into a  $t_2$ -tuple of base- $w$  words  $\mathcal{B}_2 = (b_{t_2-1}, \dots, b_0)$ . With  $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$  computed, we obtain the signature

$$\alpha = (f^{b_{t_1-1}}(x_{t_1-1}), \dots, f^{b_0}(x_0)).$$

*Signature Verification.* To assert the correctness of the signature  $\alpha$ , each of its blocks  $\alpha_i$  needs to be verified separately by computing the remaining applications of  $f$ . The signature verifies correctly if

$$Y = (f^{2^w-1-b_{t_1-1}}(\alpha_{t_1-1}), \dots, f^{2^w-1-b_0}(\alpha_0)).$$

In Figure 1 we illustrate the signature generation and its verification. It is visible that computing a signature block is a step necessarily after the key generation. Hence, the key and signature generation may only be computed by whoever has the secret signature key  $X$ .

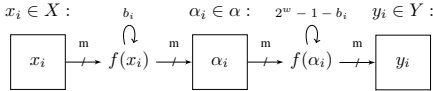


Fig. 1. WOTS signature and verification steps for an index  $i \in \{0, \dots, t-1\}$ .

### III. CHECKSUM PADDING

In this section, we propose to pad unused bits, reserved for  $c$ , with 1. Recall that  $c = \sum_{i=t-t_1}^{t-1} (2^w - 1 - b_i)$ . Define  $c_{max}$  and  $c_{min}$  as the greatest and smallest possible values of  $c$ . These situations happen when,  $\forall b \in \mathcal{B}_1$ ,  $b = 0$  or  $b = 2^w - 1$ , respectively. Hence,  $c_{max} = \sum_{i=t-t_1}^{t-1} (2^w - 1 - 0) = t_1(2^w - 1)$  and  $c_{min} = \sum_{i=t-t_1}^{t-1} (2^w - 1 - 2^w - 1) = 0$ . Additionally, the number of bits needed to represent all possible values of  $c$  is given by  $n_c = \lceil \log_2 c_{max} \rceil = \lceil \log_2 t_1(2^w - 1) \rceil$  and the number of blocks to accommodate  $c$  is given by  $t_2$ , defined in Section II.

In general, the number of bits reserved for  $c$ , that is,  $t_2w$  bits, is greater than the number of bits actually required for their representation. This difference occurs because an integer number of blocks of size  $w$  is used to accommodate  $c$ . The number of unused bits is defined as  $n_u = t_2w - n_c$ .

We note that  $w$  and  $n_u$  grow together, as seen in Table I. It groups several parameters for WOTS, how they affect  $n_u$ , and presents how these parameter groups benefit from this optimization. Odd values for  $w$  show no abnormal behavior

and are suppressed for simplicity, although there are some combinations where no padding is needed, such as  $w = 3, m = 128$  and  $w = 7, m = 512$ . Finally, we redefine  $c$  to

$$c^p = c + (2^{n_u} - 1)2^{n_c} = c + 2^{t_2w} - 2^{n_c}.$$

In other words, during the signature generation step, we fill the unused  $n_u$  bits with ones. We call the modified scheme WOTS-B.

TABLE I  
UNUSED BITS ON  $\mathcal{B}_2$  FOR VARIOUS COMBINATIONS OF  $w$  AND  $m$ .

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

### A. Security Considerations

We recall that, in Section II, the classical WOTS implementation already uses a padding of zeroes in the signature generation algorithm. Our proposal based on flipping the padding bits from zeroes to ones moves this fixed amount of iterations of  $f$  from the verification process to the signature generation. These computations have no impact in security, since they have no checksum purpose, but must be evaluated nevertheless.

### IV. TUNING $\mathcal{B}_1$

We propose a method to speed up the WOTS signature generation or verification without any modification to the original scheme. Our idea is to append a cryptographic nonce to the signed message, before generating a signature. We show in Section V that by repeating this process and searching for a suitable nonce, we can significantly reduce the cost of the signature verification, in exchange of an increased cost of the signature generation, or vice-versa. In the following, we explain the method and give statistical thresholds for the searching process.

Let  $M$  be any message,  $R \in \mathbb{N}^*$  and  $\lambda = (\lambda_0, \dots, \lambda_{R-1})$  an  $R$ -tuple of nonces. We compute all  $d_r$  such that  $d_r = \mathcal{H}(M \parallel \lambda_r)$  with  $0 \leq r \leq R$ . Recall that  $d_r$  may be split into a  $t_1$ -tuple, now defined by  $\mathcal{B}_{r,1} = (b_{r,t_1-1}, \dots, b_{r,t_1-1})$ , and let the set of summations of the integer representations of the elements in these tuples be defined by  $S = \{\sum_{b \in \mathcal{B}_{r,1}} : 0 \leq r \leq R\}$ .

Finally, we choose  $\lambda_r$  from  $\min(S)$  to obtain a faster signature generation or from  $\max(S)$  for a faster signature verification. Then, we proceed with the classical WOTS signature generation by letting  $B_1 = d_r$  or, simply by appending  $\lambda_r$  to the initial message before running the signature algorithm.

We call the method WOTS-R. Furthermore, this proposal is inspired by [19], and therefore we observe that  $\lambda$  could be replaced by the trivial set  $\{0, \dots, R\}$ .

#### A. Finding a threshold for $R$

In this proposal,  $R$  is a statistical parameter that represents the sufficient number of hashes needed to find an adequate summation in  $S$ . Intuitively, larger  $R$  should produce better results. However, this choice translates to a higher cost for signature generation. We show that there is a suitable threshold for  $R$ , depending on the required optimization.

Consider  $\mu$  as the function that calculates the arithmetic mean of a set of integers. If we assume that  $\forall b \in B_{r,1}, 0 \leq b \leq 2^w - 1$  follows a uniform distribution, then by repeatedly calculating  $\mu(B_{r,1})$ , we expect the average  $\mu' = \mu(\mu(B_{r,1})) = 2^{w-1}$ , for  $0 \leq r \leq R$ . Hence, the distribution of the averages  $\mu(B_{r,1})$  should follow a normal distribution.

We experiment with  $w = 16$ ,  $m = 256$  and  $\mathcal{H} = \text{SHA-256}$ , by computing  $\mu(B_{r,1})$  with  $0 \leq r < 2^{16}$ , and indeed verify that the distribution of the averages follows a normal distribution. Figure 2 is a graphical representation of this behavior. Then, by using the standard normal table (Z-table) with standard deviation  $\sigma$ , we have

$$P(\mu(B_{r,1}) > \mu' + \sigma) = 0.1587,$$

$$P(\mu(B_{r,1}) > \mu' + 2\sigma) = 0.0228,$$

$$P(\mu(B_{r,1}) > \mu' + 3\sigma) = 0.0013.$$

In Figure 3, we plot the chance of finding  $\mu(B_{r,1})$  inside these three intervals, with respect to  $R$ . We use the binomial distribution and distinguish three thresholds as suggestions for values of  $R$ : {25, 200, 3500}. Each value yields a probability of 99.9% of finding  $\mu(B_{r,1})$  in the intervals previously mentioned, respectively.

#### B. Security Considerations

Our proposal makes no attempt to modify the underlying classical WOTS mechanisms. Hence, we discuss the impact of appending a cryptographic nonce  $\lambda_r$  to a message  $M$  before hashing it. Recall that we must find a suitable hash  $d_r = \mathcal{H}(M \parallel \lambda_r)$  such that it produces a maximized or minimized sum of  $B_{r,1}$ . In other words, high-order bits of various blocks in  $B_{r,1}$  have high probability of being fixed, making partial hash collision attacks more susceptible.

This behavior may be exploited through differential cryptanalysis on the fixed bits for the Merkle-Damgård construction, recently put in practice to generate the first practical collision for SHA-1 [22]. Such an attack could present a threat to our proposal, thus requiring the use of cryptographic hash functions which are second preimage resilient, such as SHA-256. We leave the remaining security considerations for the scheme to be taken from [18].

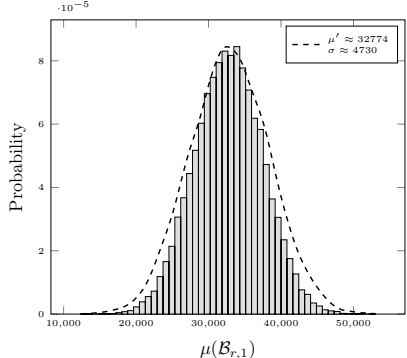


Fig. 2. Normalized histogram of  $\mu(B_{r,1})$ , with 50 bins.

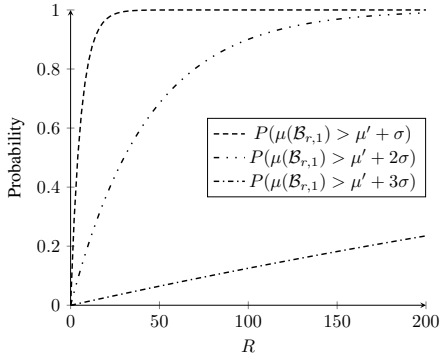


Fig. 3. Thresholds for  $R$ .

## V. EXPERIMENTS

As a proof-of-concept, we compare the number of iterations of a function  $f$  throughout the entire execution of the digital signature schemes. Again, we denote the four variants tested as WOTS for the classical scheme described in Section II, WOTS-B for the variant described in Section III, WOTS-R is the scheme described in Section IV and WOTS-BR merges the characteristics of the latter two. Considering the discussion on Subsection IV-A, sufficient values of  $R$  were chosen according to the usual values of the Winternitz parameter  $w$ .

In Table II we give the average number of iterations of  $f$  needed to verify a signature. We experiment with  $2^{14}$  executions of the verification step for the proposed schemes, with  $\mathcal{H} = \text{SHA-256}$ ,  $m = 256$  and base-16 messages of  $2^{10}$  length generated through `/dev/urandom`. To better understand the effect of each proposal individually, we also

compute the average of  $\mathcal{B}_1$  and  $\mathcal{B}_2$  separately.

Note that  $\mu(\mathcal{B}_2)$  is affected negatively by WOTS-R, since maximizing the sum of elements in  $\mathcal{B}_1$  has a direct impact on the calculation of the checksum. However, this difference does not heavily impact the overall gains achieved in  $\mu(\mathcal{B}_1)$ . Furthermore, by using WOTS-BR, this behavior is greatly mitigated and may speed up the signature generation or verification steps up to a factor of half in a best-case scenario.

TABLE II  
NUMBER OF ITERATIONS OF  $f$  ON THE VERIFICATION STEP FOR THE PROPOSED SCHEMES WHEN  $\max(S)$  IS CHOSEN.

$w$	$R$	SCHEME	$\mu(\mathcal{B}_1)$	$\mu(\mathcal{B}_2)$	$\mu(\mathcal{B})$
4	-	WOTS	479.93	25.88	505.80
		WOTS-B		12.25	492.18
	25	WOTS-R	407.81	27.49	435.29
		WOTS-BR		13.49	421.29
	200	WOTS-R		29.23	408.31
		WOTS-BR	379.08	15.23	394.31
3500	WOTS-R		31.14	380.02	
	WOTS-BR	348.88	17.14	366.02	
8	-	WOTS	4081.84	368.43	4450.27
		WOTS-B		136.25	4218.08
	25	WOTS-R		370.56	3632.95
		WOTS-BR	3262.39	130.56	3392.95
	200	WOTS-R		372.13	3312.76
		WOTS-BR	2940.63	132.13	3072.76
3500	WOTS-R		374.75	2979.24	
	WOTS-BR	2604.49	134.75	2739.24	
16	-	WOTS	525120.63	98231.81	623352.44
		WOTS-B		32707.82	557828.45
	25	WOTS-R		98225.54	474775.78
		WOTS-BR	376550.24	32697.52	409247.76
	200	WOTS-R		98850.06	418340.08
		WOTS-BR	319490.02	33321.91	352811.94
3500	WOTS-R		101022.36	363324.28	
	WOTS-BR	262301.92	35492.57	297794.48	

In general, we observe a reduction of roughly  $2^w$  iterations of  $f$  with WOTS-B alone. In the case of WOTS-R, we obtain a reduction of up to 25% for  $w = 4$ , 33% for  $w = 8$  and 42% for  $w = 16$ . By merging both schemes together, we improve these results to 28%, 39% and 52%, respectively.

The aforementioned reductions translate to an increase of similar magnitude on the signature generation. For example, according to Section II and by letting  $w = 4$ , then  $t = 67$  and the total number of iterations of  $f$  for signature generation and verification is equal to  $t \times 2^w = 1072$ . Our results show that, when  $R = 25$  with WOTS-BR, we can decrease the number of iterations of  $f$  during the signature verification from approximately 506 to 421, in average.

Avoiding these 85 iterations of  $f$  during the signature verification means that we must now calculate these on the signature generation. In other words, this amounts to a 16.8% speedup for the verification step at the cost of a 15% slowdown during the signature generation, without taking the computation of  $R$  into account.

We can substantially increase this trade-off by letting  $R = 200$  or  $R = 3500$ , when signature generation time is not constrained. Otherwise, for small values of  $w$ , the number of hashes used for WOTS-R might not be an attractive choice.

Hence, such values can be better used with greater values of  $w$ , where computing hundreds of hash functions is negligible compared to  $t \times 2^w$ .

#### A. Impact on Merkle signature schemes

Our proposal has significant results for hash-based schemes that make use of Merkle tree structures. We test WOTS-B, WOTS-R and WOTS-BR with the public domain<sup>1</sup> reference implementation of XMSS from the IETF Internet Draft [9].

We patch the reference implementation by modifying the padding process inside the WOTS+ signing algorithm, according to Section III, denoting this modification as XMSS-B. Furthermore, by choosing  $R$  with the method described in Section IV, we achieve up to 32% of speedup when benchmarking the verification step for XMSS. We call this variant XMSS-R. Additionally, when these optimizations are used together, we call the resulting scheme XMSS-BR.

Table III shows the average run time of  $2^{14}$  signatures for each scheme, including the computation of  $R$  for XMSS-R. We use the recommended value of  $w = 4$ , and additionally,  $w = 8$ . For greater values of  $w$  (e.g. 16), it is widely known that the XMSS key generation algorithm is too slow and unpractical. Hence, this value is omitted from the results. Furthermore, we experiment with both  $\max(S)$  and  $\min(S)$  to demonstrate the impact of our schemes when choosing to optimize signature verification or generation, respectively. In the case of  $\min(S)$ , XMSS-B and XMSS-BR are not considered, since WOTS' default padding already optimizes signature generation.

Our results in Table III show that, for  $\max(S)$ , any value of  $R$  improves the signature verification run time, with the associated cost on signature generation. Evidently, this behavior is suppressed with greater values of  $w$ . However, for  $\min(S)$ , not every value of  $R$  may be chosen. In the case of  $w = 4$  and  $R = 25$ , we obtain a speedup of 9.4% for the signature generation process, while when  $R = 200$  or  $R = 3500$ , both processes present a slower run time. The same reasoning applies to  $w = 8$ , where one should only choose  $R = 25$  or  $R = 200$ .

## VI. FINAL REMARKS

In this paper, we propose methods to adjust WOTS for faster signature verification at the cost of a slower signature generation, or vice-versa. Our first proposal, WOTS-B, consists of flipping the checksum padding bits, thus improving signature verification run time. In our second proposal, WOTS-R, we present a choice to to maximize or minimize the base- $w$  blocks constructed from the message digest. The former is used to speed up signature verification, whereas the latter speeds up signature generation or verification.

We experiment with the classical Winternitz scheme and a state-of-the-art Merkle-based signature scheme, XMSS. We obtain an improvement factor of up to two when  $w = 16$ , for the WOTS signature verification process, with regards to the number of iterations of  $f$ . Also, by applying our proposals

<sup>1</sup><https://github.com/joostrijnveld/xmss-reference/>

TABLE III  
SIGNATURE GENERATION AND VERIFICATION RUN TIMES (IN MS) FOR THE  
PROPOSED SCHEMES.

	$w$	$R$	SCHEME	SIG. TIME	VER. TIME
$max(S)$	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		
$min(S)$	4	-	XMSS	0.971	0.746
			XMSS-B	1.079	0.832
		25	XMSS-R	1.806	0.885
			XMSS-BR	3.393	0.898
		200	XMSS-R	7.819	5.731
			XMSS-BR	6.672	6.553
	3500	XMSS-R	6.472	6.982	
		XMSS-BR	8.488	7.435	

Relevant computer specifications are as follows: 8 GB of DDR3 RAM @ 1333MHz, Intel Core i5-4570 @ 3.2GHz and gcc 7.3.0. Base commit for the modifications: 05dac989c40349ad54f4ee3b563b85131b95332.

to XMSS, we reduce the run time for the verification step by 22% when  $w = 4$  and 36% when  $w = 8$ . Finally, when improving signature generation, for  $w = 4$  and  $w = 8$ , we obtain speedups of 9.4% and 17.2%, respectively.

#### A. Future Works

Throughout our work, we consider using  $min(S)$  and  $max(S)$  to find cryptographic nonces that speed up the WOTS scheme. However, this may not be the best approach. We believe that the criteria for choosing  $\lambda_r$  can be improved, by searching for  $B_{r,1}$  with elements of similar magnitude, in addition to the original proposal of  $\mu(B_{r,1}) > \mu'$ . Parallel implementations can exploit this trait and allow faster computations of  $f$ .

#### REFERENCES

- [1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. [Online]. Available: <https://arxiv.org/abs/quant-ph/9508027v2>
- [2] D. J. Bernstein, J. Buchmann, and E. Dahmen, *Post Quantum Cryptography*, 1st ed. Springer Publishing Company, Incorporated, 2008.
- [3] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, Sep. 2017.
- [4] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," in *STOC '90: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, H. Ortiz, Ed., May 1990, pp. 387–394.
- [5] J. Katz and C.-Y. Koo, "On constructing universal one-way hash functions from arbitrary one-way functions," *Cryptology ePrint Archive*, Report 2005/328, Sep. 2005. [Online]. Available: <https://eprint.iacr.org/2005/328>
- [6] D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kibbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijnveld, and P. Schwabe, "SPHINCS<sup>+</sup> – Submission to the NIST post-quantum project," Dec. 2017. [Online]. Available: <https://sphincs.org/data/sphincs+specification.pdf>
- [7] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures," in *Advances in Cryptology – EUROCRYPT 2015*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056, Apr. 2015, pp. 368–397. [Online]. Available: <https://eprint.iacr.org/2014/795>
- [8] G. Endignoux, "Design and implementation of a post-quantum hash-based cryptographic signature scheme," Master's thesis, École polytechnique fédérale de Lausanne, Jul. 2017. [Online]. Available: <https://gendignoux.com/assets/pdf/2017-07-master-thesis-endignoux-report.pdf>
- [9] A. Hülsing, D. Butin, S.-L. Gazdag, J. Rijnveld, and A. Mohaisen, "XMSS: Extended Hash-Based Signatures," Internet Engineering Task Force, Internet-Draft draft-irtf-cfrg-xmss-hash-based-signatures-12, Jan. 2018, Work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-xmss-hash-based-signatures-12>
- [10] D. A. McGrew, M. Curcio, and S. Fluhrer, "Hash-Based Signatures," Internet Engineering Task Force, Internet-Draft draft-mcgrew-hash-signs-08, Oct. 2017, Work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-mcgrew-hash-signs-08>
- [11] C. Dodt, N. P. Smart, and M. Stam, "Hash based digital signature schemes," in *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, ser. LNCS, N. Smart, Ed., vol. 3796, Dec. 2005, pp. 96–115.
- [12] P. Lafrance, "Digital signature schemes based on hash functions," Master's thesis, University of Waterloo, Apr. 2017. [Online]. Available: <https://uwaterloo.ca/waterloo.ca/handle/10012/11679>
- [13] L. Lamport, "Constructing digital signatures from a one-way function," SRI International Palo Alto, Technical Report CSL-98, Oct. 1979. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/>
- [14] R. C. Merkle, "A certified digital signature," in *Advances in Cryptology – CRYPTO '89*, ser. LNCS, G. Brassard, Ed., vol. 435, Aug. 1989, pp. 218–238.
- [15] J. Buchmann, L. C. C. García, E. Dahmen, M. Döring, and E. Klintsevich, "CMSS: An improved merkle signature scheme," in *Progress in Cryptology – INDOCRYPT 2006*, ser. LNCS, R. Barua and T. Lange, Eds., vol. 4329, Dec. 2006, pp. 349–363. [Online]. Available: <https://eprint.iacr.org/2006/320>
- [16] J. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume, "Merkle signatures with virtually unlimited signature capacity," in *Applied Cryptography and Network Security*, ser. LNCS, J. Katz and M. Yung, Eds., vol. 4521, Jun. 2007, pp. 31–45.
- [17] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS - a practical forward secure signature scheme based on minimal security assumptions," in *Post-Quantum Cryptography*, ser. LNCS, B.-Y. Yang, Ed., vol. 7071, Nov. 2011, pp. 117–129. [Online]. Available: <https://eprint.iacr.org/2011/484>
- [18] A. Hülsing, "W-OTS<sup>+</sup> – shorter signatures for hash-based signature schemes," in *Progress in Cryptology – AFRICRYPT 2013*, ser. LNCS, A. Youssef, A. Nitaj, and A. E. Hassanien, Eds., vol. 7918, Jun. 2013, pp. 173–188. [Online]. Available: <https://eprint.iacr.org/2017/965>
- [19] R. Steinwandt and V. L. Villányi, "A one-time signature using run-length encoding," *Information Processing Letters*, vol. 108, no. 4, pp. 179–185, Oct. 2008.
- [20] J. P. Cruz, Y. Yatanı, and Y. Kaji, "Constant-sum fingerprinting for Winternitz one-time signature," in *2016 International Symposium on Information Theory and Its Applications (ISITA)*, Oct. 2016, pp. 703–707.
- [21] G. C. C. F. Pereira, C. Puoduzis, and P. S. L. M. Barreto, "Shorter hash-based signatures," *Journal of Systems and Software*, vol. 116, pp. 95–100, Jun. 2016.
- [22] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full SHA-1," in *Advances in Cryptology – CRYPTO 2017*, ser. LNCS, J. Katz and H. Shacham, Eds., vol. 10401, Jul. 2017, pp. 570–596. [Online]. Available: <https://eprint.iacr.org/2017/190>