XXVIII Congresso {virtual} de Iniciação Científica da Unicamp 2020

Assinaturas digitais pós-quânticas baseadas em hash e voltadas para blockchains

Aluno: José Augusto A. G. de M. Braga Professor: Marco A. A. Henriques
Faculdade de Engenharia Elétrica e de Computação
joseaugusto.braga@gmail.com maah@unicamp.br

1. Introdução

Blockchain é uma arquitetura que está ganhando o interesse de empresas e governos pela sua capacidade de armazenar dados de uma maneira confiável, distribuída e robusta a fraudes [Wood 2014]. Os usuários da blockchain são capazes de fazer transações (financeiras ou não) que não podem ser desfeitas e nem ter sua autoria negada e que outros usuários podem verificar sua autenticidade e a integridade. Não é possível repeti-las ou alterá-las, principalmente devido ao uso de técnicas de encadeamento de hashes e de assinatura digital.

Os principais esquemas de assinatura digital como ECDSA [Johnson, Menezes and Vanstone 2001] e RSA [Rivest Shamir Adleman 1978] são resistentes à ataques por computadores convencionais. Porém, essas assinaturas se tornarão inseguras com o surgimento de um computador quântico de alto desempenho capaz de executar o algoritmo de Shor [Shor 1994]. Em vista dos avanços na computação quântica [Arute, Arya, Babbush 2019], a comunidade científica viu como necessário o desenvolvimento e a implementação de assinaturas resistentes a um computador quântico, as chamadas assinaturas pós-quânticas. Um exemplo desse esforço é aquele realizado pelo NIST (*National Institute of Standards and Technology*) que realizou uma chamada [Chen 2016] e recebeu diversas propostas de algoritmos de assinatura e também de criptografia resistentes ao computador quântico, que estão sendo avaliados para, se aprovados, constituírem os padrões de criptografia pós-quântica recomendados pelo órgão.

Dentre as diversas propostas de assinaturas pós-quânticas [Bernstein and Lange 2017], as baseadas em função hash ganharam destaque pela sua simplicidade e segurança já conhecidas. Entretanto, as chaves e assinaturas gerados por esses algoritmos exigem mais espaço para serem armazenadas e tempo maior de criação e verificação que as assinaturas tradicionais. Essas alterações podem interferir fortemente no desempenho de uma blockchain, já que ela requer um grande número de assinaturas e tem que verificar as mesmas sempre que necessário.

Há vários tipos de assinaturas baseadas em hash na literatura, desde sua proposta original em 1979 por Leslie Lamport [Lamport 1979] até as variantes mais sofisticadas como Sphincs+ [Bernstein, Hopwood 2015], passando por alternativas como LMS (Leighton-Micali Hash-Based Signature) [Leighton and Micali 1995] e XMSS (eXtended Merkle Signature Scheme) [Hülsing 2015]. Esse trabalho focou e comparou os dois últimos esquemas, pelo fato de serem ambos *stateful* (isto é, requerem um controle rigoroso de quais chaves já foram usadas) e de estarem detalhadamente documentados em propostas de padrões para a Internet (RFC - *Requests for Comments* 8391 [Huelsing 2018] e 8554 [McGrew 2019]). Essa comparação tem como objetivo avaliar as melhores condições para o uso dessas assinaturas em aplicações nas quais o tamanho e o tempo de verificação precisam ser minimizados, como em blockchains.

2. Uso das assinaturas digitais nas blockchains

A blockchain é um sistema também conhecido como livro-razão onde são registradas transações que os usuários fazem entre si de modo que possam ser verificadas e comprovadas por qualquer outro usuário. Múltiplas cópias da blockchain são mantidas por servidores especiais chamados de nós de mineração, que são os responsáveis por criar os blocos que contêm as transações.

Há blockchains permissionadas e não permissionadas: as permissionadas, também chamadas de privadas, são aquelas em que os usuários necessitam de um convite para participar; já as não-permissionadas, ou públicas, são aquelas em que qualquer usuário pode entrar e participar. Este trabalho foca as blockchains públicas, como, as das criptomoedas Bitcoin [Nakamoto 2008] e Ethereum [Wood 2014].

Na mais famosa blockchain, a que sustenta a criptomoeda Bitcoin, a formação do bloco ocorre da seguinte maneira: cada nó de mineração escolhe algumas das transações divulgadas na rede e que estão aguardando validação, verifica a validade delas (incluindo suas assinaturas digitais) e as adiciona ao seu bloco de transações em construção. Todos os nós que estão tentando produzir um bloco precisam passar pelo desafio do mecanismo de consenso da rede (PoW: *Proof of Work*). Somente o nó que for o mais rápido em atender as exigências desse desafio e receber a confirmação dos demais nós da rede será recompensado pelo bloco produzido e as informações inseridas no bloco serão ratificadas e imutáveis à medida que outros blocos forem anexados à blockchain. É importante notar que para que esse sistema funcione é necessário que a maioria dos nós seja honesta; caso contrário, a blockchain pode sofrer ataques a partir dos nós maliciosos que serão maioria.

Esse modelo de publicar as transações para todos presentes na rede possibilitaria que alguém mal intencionado alterasse, replicasse ou até forjasse uma transação se não houvesse garantias criptográficas. Para que isso não seja possível, as blockchains utilizam, entre outras técnicas, a assinatura digital, que impede que pessoas que não tenham a posse da chave privada façam transações com assinaturas válidas ou alterem uma transação já assinada. Para verificar a assinatura anexada a uma transação, é necessário ter acesso à chave pública que faz par com a chave privada que gerou a assinatura. Se a transação, sua assinatura ou a chave for alterada, a verificação não consegue validar a assinatura. Deve ser ressaltado que o encadeamento de blocos exigirá do atacante um esforço extra, pois além de atacar e alterar a assinatura de um bloco, ele precisará ter poder computacional suficiente para poder alterar os hashes deste bloco e de todos os sucessores de forma mais rápida que os sucessores são adicionados à cadeia.

Nas principais blockchains, o tamanho do bloco é limitado, o que interfere diretamente na quantidade de transações que podem ser armazenadas por bloco e no custo destas transações. Com isso, é importante que as transações e as suas assinaturas sejam as menores possíveis para maximizar a quantidade de transações aprovadas em um único bloco. Outro ponto importante é o tempo de verificação das assinaturas, uma vez que a operação de verificar será executada pela maioria dos elementos da blockchain e quanto mais rápido for, melhor para o desempenho da blockchain.

Neste trabalho discutimos a alteração da assinatura digital presente em blockchains que já podem estar ativas. Neste caso, é necessário analisar, ainda que de forma breve, os desafios de tal alteração. Será necessário acrescentar um novo esquema de assinatura

e fazer um acordo para uma data em que o mesmo começará a ser aceito. Mesmo que o método de assinatura anterior se torne inseguro, as assinaturas pré-existentes precisarão continuar válidas, até que as transações garantidas por elas sejam refeitas com novas e mais seguras assinaturas pós-quânticas.

3. Ameaças do computador quântico para as assinaturas digitais tradicionais

A segurança da assinatura digital está associada à dificuldade de deduzir a chave privada a partir da chave pública. Nos principais esquemas de assinaturas atuais, esta segurança se baseia na fatoração de inteiros grandes em dois números primos, no caso do RSA, ou no cálculo do logaritmo discreto sobre curvas elípticas, no caso do ECDSA, problemas que os computadores tradicionais não conseguem resolver em tempo hábil para números grandes.

Em 1994, Peter Shor [Shor 1994] desenvolveu um algoritmo para computadores quânticos que é capaz de resolver a fatoração de inteiros grandes e o cálculo de logaritmo discreto em tempo polinomial. Assim, com um computador quântico que seja capaz de executar esse algoritmo com números grandes, todos os pares de chaves dos principais esquemas tradicionais estariam suscetíveis à quebra da segurança com o cálculo da chave privada a partir da chave pública.

As assinaturas pós-quânticas baseiam sua segurança em problemas que não têm solução conhecida em tempo polinomial, tanto em computadores convencionais como quânticos. Logo, elas são resistentes a qualquer ataque conhecido. Há diversas propostas para a implementação destas assinaturas, como as baseadas em reticulados, em códigos, em hash, entre outras. Cada uma tem suas vantagens e desvantagens e, no caso das baseadas em hash, há a vantagem de se usar funções hash que são simples, bem conhecidas e com segurança comprovada.

4. Assinaturas baseadas em hash

As assinaturas baseadas em hash (HBS - *Hash-Based Signature*) foram propostas em 1979 por Leslie Lamport [Lamport 1979], com um par de chaves de uso único (OTS - *One-Time Signature*). Porém elas só ganharam destaque na década de 2000 como uma possível assinatura pós-quântica e foram desenvolvidos vários estudos para aumentar a quantidade de assinaturas por chave. Os principais esquemas baseados nessas assinaturas utilizam a árvore de Merkle [Merkle 1979], permitindo a união de várias chaves OTS, que são verificáveis por elementos da árvore e por uma chave pública que fica na sua raiz.

4.1. Segurança das assinaturas baseadas em hash

A segurança desse tipo de assinatura está associada à função hash utilizada para construção da árvore. Para que a assinatura seja segura, é necessário o uso de uma função hash com boas propriedades criptográficas, isto é:.

- resistência de pré-imagem: dada um hash h, deve ser muito difícil encontrar uma mensagem m tal que hash(m)=h;
- resistência de segunda pré-imagem: dada uma mensagem m_a deve ser muito difícil encontrar outra mensagem m_b tal que hash(m_a)=hash(m_b);
- resistência à colisão: deve ser muito difícil encontrar duas mensagens m_a e m_b tal que hash(m_a)=hash(m_b).

Há várias funções hash conhecidas que contam com tais propriedades atestadas por muitos estudos já consolidados, como as famílias de funções SHA-2 e SHA-3.

4.2. Árvore de Merkle

A Árvore de Merkle é uma árvore binária cuja geração de um nó pai é baseada na aplicação de uma função hash sobre a concatenação dos dois nós filhos. A quantidade de folhas presentes nessa árvore é uma potência de base dois e expoente igual à altura. A Fig. 1 mostra um exemplo de uma Árvore de Merkle de altura dois, na qual os quatro nós das folhas podem ser representados pelo nó raiz.

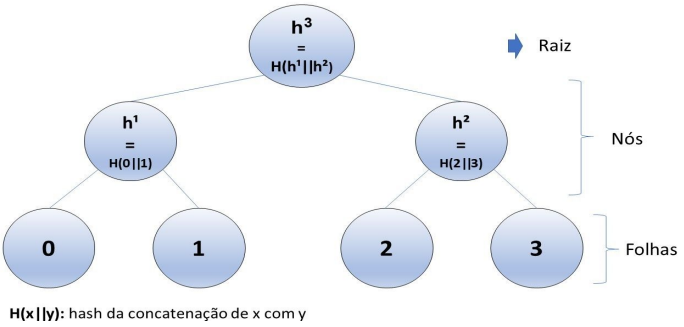

Figura 1: Árvore de Merkle com quatro folhas requer o cálculo de três funções hash.

4.3. Assinaturas únicas OTS (One-Time Signature)

As chaves OTS (*One-Time Signature*) comumente utilizadas são baseadas no esquema proposto por Lamport [Lamport 1979], como é o caso das assinaturas WOTS [Merkle 1989] ou WOTS+ [Hülsing 2013]. Nesses esquemas, a chave privada é um conjunto de L números gerados aleatoriamente. A chave pública se obtém após aplicar 2^w-1 vezes uma função hash sobre cada item da chave privada, formando um outro grupo de L elementos (chave pública), onde w $\in$ N é chamado de parâmetro de Winternitz.

A assinatura por esse par de chaves é, basicamente, uma combinação dos valores intermediários da função hash entre a chave pública e a chave privada. Para assinar será aplicada a mesma função hash sobre cada elemento da chave privada X vezes, onde X é definido pela mensagem e $1 < X < 2^w - 1$. Nota-se que a assinatura terá L elementos também. Para verificar a assinatura, é necessário aplicar a função hash $2^w - 1 - X$ vezes em cada elemento da assinatura. Se, após essa operação, o resultado for igual à chave pública, então a assinatura é válida. Uma importante propriedade deste esquema de assinaturas é que a chave privada é revelada quando se aplica a chave pública para verificar a assinatura, não podendo esse par ser usado novamente e exigindo tantos pares de chaves quantas forem as assinaturas a serem feitas.

Para unir diversas chaves OTS (One-Time Signature) é necessário o uso de uma Árvore de Merkle na qual cada folha é o hash da chave pública de um dado par de chaves OTS. Esta árvore é uma forma de representar um grande conjunto de chaves públicas OTS por uma única chave pública, sendo que a árvore não substitui tais chaves e é necessário armazenar as mesmas para uso em verificações. Com o uso da árvore, é necessário conhecer as chaves OTS já usadas, o que é feito por um contador que indica a próxima chave a utilizar,

4.4. Assinatura e verificação

A assinatura nos esquemas baseados em hash é composta por duas partes: a assinatura feita por uma chave OTS e o caminho de autenticação da árvore de Merkle. A assinatura OTS foi descrita na seção anterior. O caminho de autenticação consiste nos nós que permitem a construção parcial da árvore até a raiz, sendo necessário aplicar as regras de construção do nó. A Fig. 2 é um exemplo do caminho de autenticação, onde os nós em destaque são o caminho de autenticação para a folha 3.

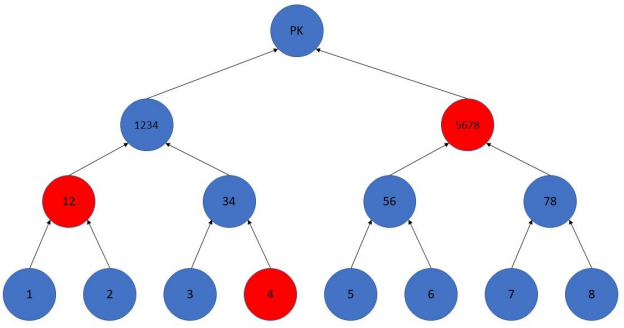

Figura 2: Exemplo de caminho de autenticação para folha 3

Para verificar uma assinatura nos esquemas baseados em hash é necessário validar a assinatura OTS presente e verificar se é possível chegar na mesma raiz da árvore com o caminho de autenticação, seguindo as regras de geração dos nós. Usando com exemplo a Fig. 2, é necessário validar a assinatura gerada pela chave OTS correspondente a folha 3 e gerar o nós 34, 1234 e PK utilizando as folhas 3 e 4 e os nós 12 e 5678.

4.5. Esquemas de assinatura *stateful* e *stateless*

Devido ao uso de chaves OTS e à impossibilidade de se reutilizar um par de chaves, é necessário manter um registro preciso de quais folhas, ou seja, quais chaves foram utilizadas para manter a segurança das assinaturas passadas e futuras. Os esquemas que têm essa necessidade são chamados de *stateful* e normalmente contêm um índice que indica a próxima folha a ser utilizada. Como as folhas, o número de assinaturas possíveis para uma dada árvore também é finito e é igual à quantidade de folhas.

Há também os esquemas de assinatura *stateless*, nos quais não é necessário ter um registro das chaves já utilizadas. Neste caso são construídas árvores muito grandes e cada folha representa apenas uma parte da chave pública. Portanto, uma chave pública precisa ser formada pela combinação de várias partes menores escolhidas aleatoriamente dentre aquelas que estão atreladas às folhas da árvore. O esquema é projetado de forma que a probabilidade de se escolher exatamente as mesmas partes e na mesma ordem é desprezível para um grande número de assinaturas (2^{64} tipicamente), o que torna praticamente impossível a repetição de uma dada chave montada a partir das partes escolhidas ao acaso e não exige o rastreamento de que folha da árvore já foi usada.

4.6. Múltiplas árvores

A construção de uma Árvore de Merkle de grandes dimensões esbarra em um grande consumo de memória. Nossos experimentos foram limitados a árvores de altura máxima igual a 40. Uma maneira alternativa de se ter uma grande quantidade de folhas é usar múltiplas árvores distribuídas em camadas. A camada mais baixa se relaciona com as chaves públicas OTS, enquanto as camadas superiores são usadas para autenticar as árvores da camada inferior e formar um caminho de autenticação até a raiz. A Fig. 3 mostra de forma simplificada como é um esquema que usa múltiplas árvores distribuídas em três camadas. Esse modelo de múltiplas árvores tem a vantagem de gerar mais assinaturas sem um grande aumento no tempo de geração das chaves, pois o esforço é dividido entre diversas árvores. Porém há desvantagens no tamanho maior das assinaturas e no tempo de verificação mais longo.

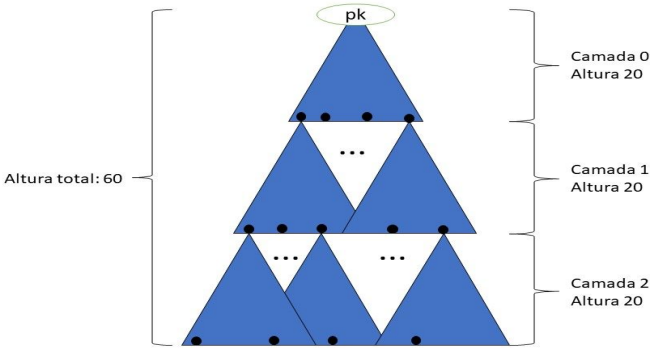

Figura 3: Esquema de múltiplas árvores de três camadas e altura 60

4.7. XMSS: eXtended Merkle Signature Scheme

O XMSS é um esquema de assinaturas *stateful* que foi um dos primeiros a utilizar a Árvore de Merkle. As funções hash recomendadas para esse esquema são SHA256, SHA512 [NIST 2015-1], SHAKE256 e SHAKE521 [NIST 2015-2]. Esse esquema tem uma pequena diferença para formação dos nós da árvore: antes de aplicar o hash sobre a concatenação dos nós filhos, ele faz um XOR desta concatenação com uma máscara, chamada de SEED, e, sobre esse resultado, aplica o hash. A chave pública desse esquema contém a raiz da árvore e a SEED pública que será utilizada para a formação dos nós para verificar a assinatura. A assinatura OTS utilizada é a WOTS+ (Winternitz One-Time Signature) e os hashes das chaves públicas serão as folhas da árvore.

Esse esquema tem duas versões: uma que utiliza uma árvore, chamada de XMSS [Hülsing 2015], e outra que pode usar múltiplas árvores, chamada XMSS^{MT} (XMSS Multi-Tree). Essa versão é caracterizada por usar árvores com os mesmos parâmetros em várias camadas [Huelsing 2013].

4.8. LMS: Leighton-Micali Hash-Based Signature

Outro esquema de assinatura baseado em hash *stateful* é o LMS. Este é mais recente que o XMSS e alega ser 4 vezes mais rápido que o XMSS [Hülsing 2015]. Porém, até o momento, só é compatível com a função hash SHA256. A criação dos seus nós é obtida pelo hash da concatenação do endereço da posição do nó na árvore e os nós filhos. A chave pública é a raiz da árvore gerada, a chave OTS é LM-OTS [McGrew 2019], com modo de operação similar ao descrito na seção 4.3, e chaves públicas OTS formam as folhas da árvore.

A sua versão de múltiplas árvores é chamada de HSS (Hierarchical Signature System) e permite que as árvores sejam diferentes entre camadas. Isso é uma vantagem pois oferece mais escolhas para o tamanho das chaves, da assinatura e do tempo necessário para gerar chaves, assinar documentos e verificar assinaturas.

5. Testes, resultados e discussões

5.1 Condições dos testes e de obtenção dos dados

Para obter os tamanhos de chaves e assinaturas e os tempos de criação de chaves, de assinatura e de verificação foram utilizados os códigos referenciados no documento RFC de cada esquema de assinatura [Huelsing 2015] [McGrew 2019]. Tais códigos foram compilados e executados em um computador com processador Intel Core i5-2430M, clock de 2.40GHz, 8GB de memória RAM e sistema operacional Ubuntu 18.04.4 LTS. Os tempos para gerar o par de chaves, assinar e verificar assinatura foram obtidos da função *clock()* presente na biblioteca *time.h* (linguagem C). Esta função retorna a quantidade de *clocks* que o programa utiliza no processador. Para obter o tempo de execução é necessário dividir o número de *clocks* por uma constante. Também foram medidos os parâmetros de ECDSA, usando a biblioteca openssl (versão 1.1.1) com a curva secp256k1 e a função hash SHA-256.

Os tempos são resultantes da média de 100 execuções do código com os mesmos parâmetros, visando diminuir os impactos que outras aplicações ou o próprio sistema operacional podem causar na medição e a imprecisão da função em contar os clocks do processador utilizados para execução do programa. (exceto quando a altura de uma árvore era igual ou superior a 20; neste caso foi calculada a média de dez execuções, já que elas levam um tempo longo para acabar).

5.2 Os impactos da otimização: versões normal e rápida dos algoritmos

Um ponto importante sobre os códigos é a opção de gerar um arquivo auxiliar durante o processo de criação das chaves para armazenar a parte da camada mais alta da árvore, sendo o tamanho desse arquivo dependente dos parâmetros de tal árvore. Esse arquivo é utilizado na operação de assinar, fornecendo os nós da árvore na camada mais próxima à raiz, dispensando o cálculo dos mesmos e diminuindo o tempo de gerar a assinatura. Porém há um custo em aumentar o tamanho da chave privada. No caso do XMSS e do XMSS^{MT}, o código oferece dois executáveis: um normal, sem otimizações, e outro chamado FAST, que armazena parte dos nós da camada superior da árvore em um arquivo auxiliar. Já o LMS não oferece estas opções de dois executáveis: o padrão do código é gerar esse arquivo auxiliar automaticamente, ou seja, a versão FAST, exceto no caso em que o usuário informa o tamanho zero para o tamanho do arquivo auxiliar, criando então uma versão normal, sem otimização.

Para a análise dos impactos desta otimização com o arquivo auxiliar, optamos pelo uso do XMSS e XMSS^{MT} pela praticidade da existência das duas versões (otimizada e não otimizada). Como esse arquivo auxiliar ajuda no processo de assinar, ele não interfere no

tempo de verificação, no tamanho da assinatura e no tamanho da chave pública; logo esses pontos não serão abordados nesta subseção. A Fig. 4 mostra os impactos desta otimização sobre o tempo para criar o par de chave e para assinar, usando um gráfico log-log. Nessa análise usamos o XMSS^{MT} com duas camadas.

Do gráfico, concluímos que o uso da versão otimizada (FAST) praticamente não tem impacto sobre o tempo de criação do par de chaves para a versão de uma camada, porém para a versão de duas camadas, o tempo quase que dobra. Analisando o tempo para assinar, nota-se uma grande diminuição com com a versão FAST, tanto na versão com uma camada quanto para duas.

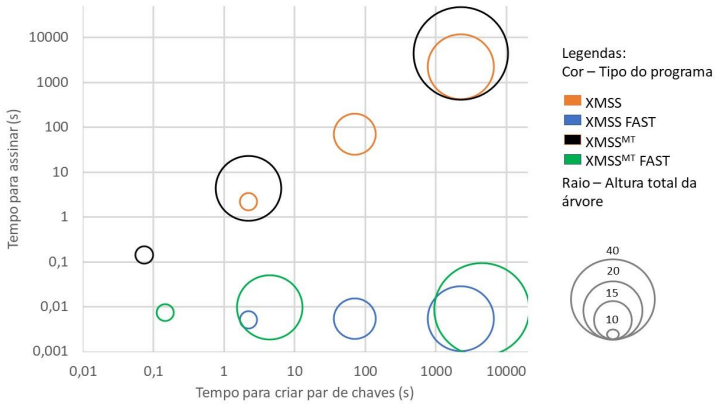

Figura 4: Impacto do arquivo auxiliar de XMSS-FAST sobre o tempo de criar chave e assinar

Para analisar o impacto da otimização sobre o tamanho da chave privada, temos a Tabela 1. Dela conclui-se que a versão FAST causa um aumento grande no tamanho das chaves privadas pois é necessário armazenar mais informações para diminuir o tempo para assinar. Outro ponto importante na versão FAST é que o uso de mais camadas implica em um aumento significativo no tamanho da chave privada, algo que não ocorre na versão normal (não otimizada).

Tabela 1: Variação do tamanho da chave privada nas diferentes versões de XMSS

Versão	Número de camadas	Altura total	Tamanho da chave privada (B)
Normal	1	10	136
		15	136
		20	136
	2	10	134
		20	135
		40	137
FAST	1	10	1377
		15	1961
		20	2577
	2	10	4153
		20	6002
		40	9604

No contexto das blockchains, o uso ou não de otimização não interfere diretamente na rede, mas afeta o usuário na questão do tamanho da chave privada que terá que armazenar e no tempo para gerar a própria assinatura. Avaliando como sendo vantajoso para o usuário que as suas transações estejam assinadas e prontas para serem validadas no menor tempo possível e já que o tamanho de chave privada não vai exigir grande volume de armazenamento, é mais vantajoso utilizar a versão otimizada FAST.

Para a análise desta subseção utilizamos os quatro algoritmos: XMSS-FAST, XMSS^{MT}-FAST, LMS e HSS, sendo que esses últimos geraram os arquivos auxiliares automaticamente (versão FAST por default). Os algoritmos de múltiplas árvores foram definidos com duas camadas nos testes. O desempenho dos quatro algoritmos é comparado com a assinatura mais comum entre as blockchains atuais: ECDSA com uma curva elíptica de 256 bits (secp256k1), para manter o mesmo nível de segurança dos quatro algoritmos pós-quânticos.

Um dos parâmetros mais importantes para essas assinaturas é a quantidade de folhas que a árvore possui, pois elas definem quantas assinaturas poderão ser feitas. Escolhemos diversas alturas para analisar o comportamento dos algoritmos, pois a altura total é definida pela quantidade de folhas da árvore. As alturas totais escolhidas para XMSS-FAST e LMS são dez, quinze e vinte, enquanto para XMSS^{MT}-FAST e HSS são de dez, vinte e quarenta. A altura quinze não é utilizada na versão de múltiplas árvores pois este possui duas camadas e seria necessário que cada camada tivesse altura de 7,5, algo que não é possível. Já a altura quarenta não é utilizada entre os algoritmos de uma camada pois seria inviável a criação de uma árvore com essa altura.

5.3 Avaliação dos tamanhos de chaves e de assinatura e dos tempos de gerar chaves, assinar e verificar assinaturas no XMSS-FAST e LMS

A Fig. 5 mostra a relação entre o tempo de gerar o par de chaves e o tempo para assinar com diferentes configurações em um gráfico log-log. Primeiro ponto que vamos analisar é o desempenho dos esquemas pós-quânticos comparado ao do ECDSA, todos em um nível de segurança de 256 bits. Vemos que o tempo de gerar o par de chaves e para assinar dos esquemas de altura total igual a dez são parecidos com os do ECDSA, mas essas chaves são capazes de assinar apenas 2^{10} vezes. Com o aumento do número de assinaturas, e da altura total, os tempos de gerar o par de chaves e de assinar para o LMS e o HSS crescem consideravelmente. No caso do XMSS-FAST e XMSS^{MT}-FAST somente o tempo de gerar chave aumenta de maneira clara, já que o tempo de assinar é praticamente o mesmo que o ECDSA.

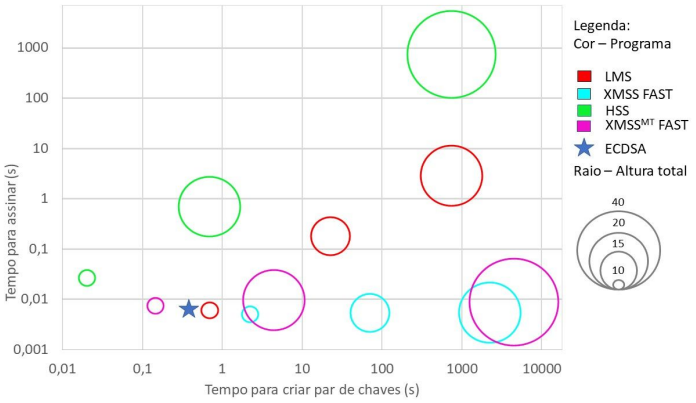

Figura 5: Tempos de criação do par de chaves e de assinatura

Agora comparando os algoritmos pós-quânticos entre si, é possível notar como a altura e a quantidade de camadas afeta os tempos de gerar o par de chaves e assinar. É possível notar que nos quatro algoritmos o aumento na altura total causa um aumento no tempo de gerar o par de chaves, pois quanto maior a altura total, maior é a quantidade de operações necessárias para gerar a chave. Porém o aumento na quantidade de camadas diminui o tempo para criar as chaves pois trabalha com árvores de altura inferior e mais fáceis de lidar. Já no tempo de assinar ocorre algo diferente: o aumento da altura total e o aumento na quantidade de camadas causa o incremento no tempo de assinar para o LMS e o HSS, porém esse comportamento não é observado no XMSS-FAST nem XMSS^{MT}-FAST devido às informações que são armazenadas no arquivo auxiliar por estes dois algoritmos, o que permite que esse tempo seja praticamente constante para os diferentes parâmetros. Comparando o desempenho geral dos algoritmos vemos que LMS e HSS têm o tempo de criar as chaves inferior comparado aos do XMSS-FAST e XMSS^{MT}-FAST, respectivamente, mas para assinar o destaque vai para as assinaturas do esquema XMSS, com o seu tempo praticamente constante para diferentes parâmetros.

A Fig.6 expõe a relação entre o tamanho da assinatura e o tempo de verificação em um gráfico linear para diferentes alturas. Nela é possível notar que, no geral, o aumento na altura total causa um ligeiro acréscimo no tamanho da assinatura e o aumento da quantidade de camadas praticamente duplica o tamanho da mesma. O tempo de verificação não tem uma grande alteração com o aumento da altura total para os algoritmos e o aumento na quantidade de camadas gera um pequeno aumento neste tempo de verificação se trocamos o LMS pelo o HSS. Porém, na troca do XMSS-FAST para o XMSS^{MT}-FAST há um aumento considerável, quase que o dobrando o tempo de verificação. Avaliando os algoritmos, o LMS se destaca em relação ao

XMSS-FAST e o HSS ao XMSS^{MT}-FAST com o seus tempos para verificar assinatura menores, mas é importante notar que todos os algoritmos tiveram uma média desse tempo na ordem de 10^{-3} segundos. Em relação ao tamanho das assinaturas, não há um algoritmo que se destaque quando se compara entre os que têm a mesma quantidade de camadas pois os tamanhos são praticamente os mesmos. Na comparação das assinaturas pós-quânticas com ECDSA é possível notar que o tempo para verificar é superior aos demais, mas o tamanho da assinatura é muito menor: 72 bytes. Percebe-se que os algoritmos pós-quânticos são mais rápidos que o clássico ECDSA, ao custo de exigirem assinaturas significativamente maiores.

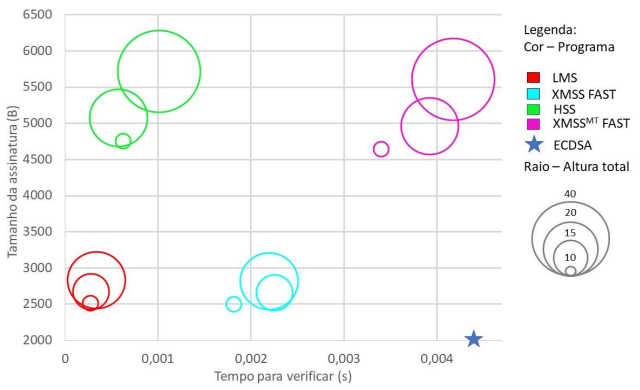

Figura 6: Relação entre a altura total das árvores, o tempo de verificação e o tamanho da assinatura

A Tabela 2 contém as informações de tamanho chaves para diferentes alturas. Nela notamos que o tamanho da chave pública não se altera com diferentes alturas totais ou camadas, pois o seu tamanho é definido pela função hash utilizada e informações necessárias para se verificar a assinatura (por exemplo, as assinaturas XMSS precisam ter na sua chave pública a raiz da árvore e a SEED pública).

Tabela 2: Tamanho das chaves para diferentes alturas

Assinaura	Número de camadas	Altura total	Tamanho da chave privada (B)	Tamahno da chave pública (B)
XMSS-FAST	1	10	1377	68
		15	1961	68
		20	2577	68
XMSS^MT-FAST	2	10	4153	68
		20	6002	68
		40	9604	68
LMS	1	10	10980	60
		15	5540	60
		20	10980	60
HSS	2	10	420	60
		20	10980	60
		40	10980	60

O tamanho da chave privada tem grande crescimento com o aumento da altura total e da quantidade de camadas utilizadas. Vemos que na maioria dos casos, o XMSS-FAST e XMSS^{MT}-FAST têm tamanho de chave privada menor que LMS e HSS, respectivamente, mas a diferença entre os tamanhos diminui quanto maior for a altura total.

Com as informações obtidas concluímos que as assinaturas XMSS-FAST, XMSS^{MT}-FAST, LMS e HSS têm um desempenho parecido nos quesitos tamanho de chave pública e de assinatura. Já nos tempos para gerar o par de chaves e verificar a assinatura o destaque fica para LMS e HSS e no tempo para gerar a assinatura as melhores escolhas são o XMSS-FAST e o XMSS^{MT}-FAST. Levando em conta que nas blockchains uma assinatura digital boa deve ter um tamanho de assinatura, chave pública e tempo de verificação os menores possíveis, LMS e HSS são melhores que XMSS-FAST e XMSS^{MT}-FAST, respectivamente, para esta aplicação pelo fato de terem o tempo para verificar inferior.

6. Conclusões

Como as principais blockchains públicas ficarão vulneráveis ao computador quântico, esse trabalho buscou avaliar o uso das principais assinaturas baseadas em hash *stateful* para substituir os atuais esquemas de assinatura baseados em curvas elípticas. As principais blockchains buscam tamanho de assinatura digitais e de chave pública menores e com baixo tempo de verificação. Analisando as principais assinaturas vemos que o LMS e o HSS otimizados tem um desempenho melhor que o XMSS-FAST e XMSS^{MT}-FAST, respectivamente, para as blockchains devido, principalmente, ao menor tempo de verificação menor e ao menor tamanho de chave pública.

Agradecimentos

Este trabalho foi parcialmente financiado com recursos do Serviço de Apoio ao Estudante (SAE) da Unicamp.

Referências

Arute, F., Arya, K., Babbush, R. *et al.* (2019) "Quantum supremacy using a programmable superconducting processor". *Nature* 574**,** 505–510

Bernstein, D., Hopwood, D., Huelsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., and Z. Wilcox-O'Hearn, (2015), "SPHINCS: Practical Stateless Hash-Based Signatures", Lecture Notes in Computer Science, Volume 9056, Advances in Cryptology - EUROCRYPT, DOI 10.1007/978-3-662-46800-5_15.

Bernstein, D., Lange, T., (2017), "Post-quantum cryptography". *Nature* 549**,** 188–194

Chen, L., Chen, L., Jordan, S., Liu, Y. K., Moody, D., Peralta, R., ... & Smith-Tone, D. (2016). *Report on post-quantum cryptography* (Vol. 12). US Department of Commerce, National Institute of Standards and Technology.

Johnson, Don & Menezes, Alfred & Vanstone, Scott. (2001). The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Sec.. 1. 36-63. 10.1007/s102070100002.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120-126.

Shor, P. W., (1994) "Algorithms for quantum computation: discrete logarithms and factoring," *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA, 1994, pp. 124-134.

Huelsing, A., Rausch, L., and J. Buchmann, (2013) "Optimal Parameters for XMSS^MT", Lecture Notes in Computer Science, Volume 8128, CD-ARES, DOI 10.1007/978-3-642-40588-4_14.

Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, (2018), "XMSS: eXtended Merkle Signature Scheme", RFC 8391

Hülsing, Andreas, (2013),"W-OTS+–shorter signatures for hash-based signature schemes. "International Conference on Cryptology in Africa. Springer, Berlin, Heidelberg

Hülsing, Andreas, et al. (2015), "XMSS: Extended hash-based signatures." Crypto Forum Research Group Internet-Draft.(2015). Draft-irtf-cfrg-xmss-hash-based-signatures-01

Lamport, Leslie. (1979) Constructing digital signatures from a one-way function. Vol. 238. Technical Report CSL-98, SRI International

Leighton, F., Micali, S. (1995): "Large probably fast and secure digital signature schemes based on secure hash functions", https://www.google.com/patents/US5432852, US Patent 5,432,852

McGrew, D., Curcio, M., and S. Fluhrer, (2019), "Leighton-Micali Hash-Based Signatures", RFC 8554

Ralph Charles Merkle. 1979. Secrecy, authentication, and public key systems. Ph.D. Dissertation. Stanford University, Stanford, CA, USA. Order Number: AAI8001972.

Merkle, R. C., (1989) "A certified digital signature," in Proceedings on Advances in Cryptology, ser. CRYPTO '89. Springer-Verlag New York, Inc.,

Nakamoto, Satoshi, (2008), "Bitcoin: A peer-to-peer electronic cash system.".

National Institute of Standards and Technology (2015-1), "Secure Hash Standard (SHS)", FIPS PUB 180-4,DOI 10.6028/NIST.FIPS.180-4.

National Institute of Standards and Technology, (2015-2) "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, 2015.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120-126.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, *151*(2014), 1-32.