Etherspace: uma abordagem Proof-of-Space na blockchain Ethereum

Diogo Castilho[2], Paulo Silva[1], João Barreto[1], and Miguel Matos[1]

[1] INESC-ID paulo.mendes.da.silva@tecnico.ulisboa.pt
joao.barreto@tecnico.ulisboa.pt miguel.marques.matos@tecnico.ulisboa.pt
[2] IST, Lisboa diogo.castilho@tecnico.ulisboa.pt

Abstract. A maior parte das *blockchains* são baseados no modelo da Bitcoin, e portanto, a sua segurança depende de provas da trabalho (*proof-of-work*, ou PoW, em inglês). Para adicionar blocos à *blockchain* é necessário que os utilizadores provem que usaram uma certa quantidade de poder computacional. As provas da trabalho fazem com que os requisitos energéticos das *blockchains* sejam muito elevados consumindo, no caso da Bitcoin, energia equivalente a um país como a Irlanda.
Este trabalho propõe uma nova implementação de *blockchain* que substitui as provas de trabalho pelas provas de espaço com o objetivo de diminuir os requisitos energéticos das *blockchains* . Numa *blockchain* que use provas de espaço, os mineiros têm de provar que estão a dedicar quantidades não triviais de memória ao protocolo. Antes de participarem no protocolo começam por realizar computações cujos resultados serão guardados na memória. Quando adicionarem blocos à *blockchain* têm de demonstrar que estão de facto a guardar corretamente o resultado dessas computações. O uso de provas de espaço em *blockchains* é um tópico recente e a maioria do trabalho nesta área é teórico, não sendo trivial como podem ser construídas soluções práticas. Este trabalho propõe uma nova implementação de *blockchain* que substitui as provas de trabalho pelas provas de espaço, realizada em cima do protocolo do Ethereum, uma das criptomoedas mais populares.

Keywords: Blockchain · Provas de espaço · Provas de trabalho · Ethereum

1 Introdução

Recentemente a *blockchain* tornou-se um tópico popular dentro e fora da comunidade de investigação. Uma *blockchain* é uma estrutura de dados que é mantida por uma rede de nós distribuída, que proporciona um registo imutável no qual apenas se pode acrescentar novos registos. Os registos estão agrupados em blocos, cada bloco contém o hash[3] criptográfico do bloco anterior, pelo que qualquer alteração à ordem dos blocos vai invalidar a cadeia. Os nós também têm de correr um protocolo de consenso para garantir que todos os nós concordam nos blocos que vão ser adicionados à *blockchain*.

[3] Neste artigo usamos hash para referir o resultado de aplicar uma função de dispersão.

O tipo mais popular de aplicações que usam *blockchains* são criptomoedas das quais Bitcoin [24] e Ethereum [4] são as mais populares. *Blockchains* que correm este tipo de aplicações costumam usar variações de um protocolo que é conhecido na literatura como Nakamoto Consensus [13]. Podemos ver o Nakamoto Consensus como a combinação de incentivos, regra de escolha de cadeia e provas de trabalho. As provas de trabalho são puzzles criptográficos que os nós têm de resolver para poderem adicionar blocos à *blockchain*. As provas de trabalho são usadas para prevenir ataques de *sybil* [16]. Estas obrigam os nós a provar que possuem recursos não falsificáveis, limitando a sua capacidade de contribuir para o protocolo. No entanto, este critério não é suficiente. Numa rede grande, dois blocos podem ser criados concorrentemente, o que vai resultar em nós com cadeias diferentes, um *fork*. Para obrigar os nós a concordarem na mesma cadeia é preciso uma regra de escolha de cadeia. No Nakamoto Consensus, os nós devem escolher a cadeia mais comprida (a cadeia que precisou de mais poder computacional para ser gerada). Cada vez que um nó acrescenta um bloco à *blockchain* recebe uma recompensa na criptomoeda respetiva. Este incentivo garante que os nós têm interesse em cumprir a regra da escolha de cadeia.

As provas de trabalho são uma das partes mais importantes do Nakamoto Consensus [13] e são usadas em muitas *blockchains* , no entanto têm algumas limitações. Desperdiçam uma quantidade significativa de energia. O'Dwyer and Malone [23] mostraram que a energia consumida pela Bitcoin é comparável à energia consumida por um país como a Irlanda. Este estudo foi efetuado em 2014, desde então o consumo de energia da Bitcoin aumentou [1]. As provas de trabalho também limitam a escalabilidade das *blockchains* . Para aceitar uma transação, a maior parte dos clientes de Bitcoin requer que pelo menos 6 blocos sejam minerados em cima do bloco que a contém. Como os blocos são adicionados em intervalos de aproximadamente 10 minutos, os utilizadores têm de esperar pelo menos uma hora antes de poderem aceitar uma transação. Atualmente, a Bitcoin apenas consegue processar 7 transações por segundo (tps), que é um número muito pequeno comparado com o Paypal, que consegue processar 115 tps e o Visa, que consegue processar 47000 tps [20, 27].

Encontrar um mecanismo para substituir as provas de trabalho nas *blockchains* é um tema importante de investigação. No entanto, esta tarefa não é fácil, pois um mecanismo que seja à prova de ataques de *sybil* requer que sejam gastos recursos, e existem pouco recursos não são facilmente falsificáveis. Uma alternativa interessante e pouco explorada às provas de trabalho são as provas de espaço [18, 8]. Em vez de dedicarem poder computacional ao protocolo os nós dedicam memória, tipicamente sob a forma de disco. As provas de espaço costumam ter duas fases. A primeira fase é a inicialização, onde o mineiro realiza computações e guarda os resultados na memória. A última fase é a execução, onde o mineiro demonstra que está a guardar corretamente o resultado das computações.

Existe um projeto de *blockchain* chamado Chia [3], que tanto quanto sabemos, é a única proposta que concretiza o uso de provas de espaço[4]. O Chia vai usar provas de espaço em conjunto com provas de tempo. Nas provas de tempo os

[4] Podemos considerar que Burstcoin [2] também utiliza este tipo de provas. Devido à falta de especificação do sistema não o vamos discutir.

mineiros provam que fizeram uma computação não paralelizável, durante uma quantidade predeterminada de tempo. As provas de tempo são uma área de investigação muito recente, que ainda tem muitas perguntas em aberto.

O uso exclusivo de provas de espaço em *blockchains* é uma área ainda pouco explorada, pelo que este trabalho explora este problema com o objetivo de reduzir o consumo de energia dos protocolos de *blockchain*. Para estudar este problema vamos implementar em cima do Ethereum uma abordagem que utilize provas de espaço, que denominámos Etherspace. Neste sistema os nós, para adicionar blocos à cadeia, vão ter de produzir provas de espaço em vez de provas de trabalho. Antes de poderem começar a minerar os nós têm de começar por realizar computações cujos resultados serão guardados na memória. Quando adicionarem blocos à *blockchain* têm de demonstrar que estão de facto a guardar corretamente o resultado dessas computações. Para isto vamos usar o mesmo modelo de provas de espaço utilizados no Chia [8]. No entanto ao contrário do Chia vamos usar exclusivamente as provas de espaço.

Na Secção 2, vamos discutir trabalho relacionado. Na Secção 3, vamos explicar a nossa abordagem, inicialmente vamos apresentar uma abordagem ingénua que usamos como ponto de partida para construir a segunda abordagem. Na Secção 4, vamos apresentar a avaliação que fizemos à nossa solução. Na Secção 5, vamos concluir este artigo com algumas considerações importantes e vamos discutir o trabalho futuro que vamos realizar.

2 Trabalho Relacionado

As provas de trabalho são puzzles criptográficos cuja solução é computacionalmente cara de encontrar, mas, por outro lado, a verificação da mesma é fácil. Inicialmente, foram propostas como uma medida de prevenção de spam e de ataques de negação de serviço [17, 10]. A sua utilidade como mecanismo de prevenção de ataques de *sybil* foi provavelmente considerada pela primeira vez por Aspnes et al [9]. No entanto, o potencial das provas de trabalho apenas foi reconhecido com o aparecimento da Bitcoin [24], a primeira criptomoeda descentralizada, agora conhecidas como criptomoedas. Este puzzle consiste em encontrar um *nonce* N tal que $H(N, B, h-1) < T$ onde H é uma função de dispersão (na Bitcoin é o SHA-256), B é o bloco atual, $h-1$ é o hash do bloco anterior e T é o objetivo de dificuldade. O número de *nonces* que é preciso experimentar até encontrar uma solução válida, cresce exponencialmente com número de bits a zero com o qual o objetivo de dificuldade começa.

Existem algumas alternativas às provas de trabalho das quais as mais conhecidas são provavelmente as provas de participação (do inglês *Proof-of-stake*). As provas de participação, inicialmente discutidas num fórum de Bitcoin [29], exigem que os utilizadores provem que são proprietários de dinheiro na criptomoeda correspondente. Neste modelo a probabilidade de um nó conseguir adicionar um bloco à cadeia é proporcional à quantidade de moeda que possuem. Apesar de recentemente terem aparecido alguns projetos com provas de segurança rigorosas [21, 19] que utilizam este paradigma, a sua segurança depende

de fortes suposições de sincronia e que uma maioria dos *stakeholders* estejam online constantemente.

O *proof-of-elapsed-time* (PoET) é um protocolo recente que utiliza ambientes de execução confiáveis (TEE) para correr uma "lotaria" para selecionar o mineiro do próximo bloco [7]. Neste protocolo cada nó usa o TEE para criar um temporizador. Quando este temporizador chegar ao fim, os outros nós podem verificar se este esperou o tempo suposto. O nó que obtiver o temporizador mais curto é escolhido para adicionar o bloco à cadeia. O PoET foi originalmente desenvolvido pela Intel e é utilizado no Hyperledger Sawtooth Lake [7]. O TEE utilizado é o *Software Guard Extensions* (SGX).

Esta abordagem tem o problema de obrigar os utilizadores a confiar em hardware de um único vendedor. Além disso, Chent et al. [14] demonstraram que um atacante apenas precisa de comprometer uma fracção de $\Theta(\frac{\log \log n}{\log n})$ nós para comprometer o sistema.

As provas de espaço foram originalmente apresentadas como uma alternativa mais ecológica às provas de trabalho [25, 18, 8]. Podemos definir uma prova de espaço como um protocolo onde um utilizador pretende provar que está a guardar dados com um tamanho N a um verificador. No contexto de uma *blockchain*, cada bloco vai ter uma prova de espaço. O utilizador é o mineiro, que antes de começar a minerar tem de correr a fase de inicialização da prova de espaço. Quando o mineiro tiver alocado a quantidade de espaço desejada pode começar a minerar. Para adicionar um bloco à cadeia vai ter de correr a fase de execução, onde vai utilizar a memória que alocou para produzir uma prova de espaço (esta prova mostra que o mineiro está a guardar corretamente a memória alocada na fase de inicialização). Esta prova é incluída no bloco. Sempre que outro nó receber este bloco vai cumprir o papel do verificador, e verificar que a prova é correta, o utilizador está a guardar corretamente dados com um tamanho N.

Existem duas abordagens significativas para as provas de espaço. A primeira é baseada em problemas de *pebbling* em grafos [18]. Esta abordagem oferece as melhores garantias que um modelo de provas de espaço pode oferecer. Um utilizador malicioso tem de usar espaço de tamanho N ou computar N vezes uma função de dispersão para que um verificador aceite a prova. No entanto existem algumas limitações que dificultam a sua adaptação num protocolo de *blockchain*. Primeiro, o tamanho da cada prova é grande, na ordem dos Mbytes. Segundo, não se consegue criar uma versão não interativa da prova. Consequentemente novos mineiros que se queiram juntar ao protocolo precisam que outros mineiros incluam nos seus blocos transações especiais de confirmação. Se todos os mineiros pararem de aceitar novas transações de confirmação, nós novos não poderão participar no protocolo de mineração.

A segunda abordagem para provas de espaço é baseada em inverter funções aleatórias [8]. Neste modelo, o verificador envia durante a inicialização a descrição de uma função $g_f : [N] \rightarrow [N]$ a um utilizador, onde $g : [N] \times [N] \rightarrow [N]$ é uma função aleatória e $f : [N] \rightarrow [N]$ é uma permutação aleatória e $g_f(x) = g(x, x')$ com a condição de $f(x) = \pi(f(x'))$ [5] para qualquer involução π. Depois de receber a descrição da função g_f, o utilizador vai computar a tabela

[5] Se modelarmos f como uma função aleatória a condição torna-se $f(x) = f(x')$

desta função (ou seja todos os pares $(x, x', g(x, x'))$ tal que $f(x) = f(x')$), esta tabela vai ser usada para criar provas de espaço. Durante a fase de execução o verificador vai enviar ao utilizador valores aleatórios $y \in N$ aos quais o utilizador tem de responder com pares (x, x') tal que $f(x) = f(x')$ (no caso de modelarmos f com uma função aleatória) e $g(x, x') = y$. Ao aplicarmos estas provas de espaço numa *blockchain* podemos considerar que uma prova de espaço (desafio) é constituída pela inversão de número predeterminado de valores aleatórios $y \in N$. Esta abordagem tem a vantagem de ter provas com um tamanho pequeno e de se conseguir criar uma versão não interativa.

Esta construção foi criada para ser utilizada no Chia [3], um projeto em desenvolvimento para uma criptomoeda. O Chia ainda não tem nenhum relatório técnico, no entanto existem palestras nas quais se explica como o sistema funciona [15], e artigos que explicam a matemática que vão utilizar [12, 8]. Este protocolo vai usar uma combinação de provas de espaço com provas de tempo. As provas de tempo utilizam funções de atraso verificáveis (VDF) [12], funções que demoram um tempo predeterminado a computar, e não são paralelizáveis (o tempo de computação não diminui ao computar a função em paralelo). Neste modelo os mineiros de espaço adicionam os seus blocos à cadeia. Quanto melhor for a prova de espaço mais curto vai ser o tempo de computação das provas de tempo. Depois de propagados os blocos os mineiros de "tempo" vão finalizar o bloco com a prova de tempo. Esta abordagem, embora promissora, sofre do problema de as VDFs serem uma área recente com suposições não triviais e muitas perguntas em aberto. Existem duas propostas significativas, de Pietrzak [26] e Wesolowski [28]. A proposta de Pietrzak requer que os seus parâmetros públicos sejam criados por um terceiro confiável ou por um protocolo de computação *multiparty*. A construção de Wesolowski requer que a suposição *adaptive root* se mantenha [12], que é um problema pouco estudado. Além disto se um atacante conseguir implementar uma versão mais rápida destes algoritmos, pode comprometer a integridade da *blockchain* do Chia.

3 Etherspace

Nesta secção, começamos por discutir uma primeira abordagem ingénua de como utilizar as provas de espaço num protocolo de *blockchain* e onde esta abordagem falha. Depois apresentamos uma abstração que permite resolver o problema e que usamos para construir a nossa solução.

3.1 Abordagem Ingénua

De entre as duas abordagens de provas de espaço discutidas na Secção 2, decidimos escolher a baseada em inverter funções aleatórias [8]. A principal razão é que produzir estas provas de forma não interativa é fundamental em *blockchain*.

Nesta solução, antes de poder começar a minerar, o mineiro tem de alocar memória para poder produzir provas de espaço, i.e, calcular a tabela da função g_f. Para adicionar um bloco à cadeia, o mineiro utiliza os últimos n bits do hash do último bloco como desafio da prova de espaço y (o valor que ele tem de

inverter). Como a tabela da função g_f é bastante extensa existe a probabilidade que nenhum mineiro possua o par (x, x'), tal que $g_f(x) = y$, para um dado desafio y. Portanto os mineiros em vez de apresentarem exatamente a inversão (x, x') de um valor y, têm de apresentar a inversão que origina o valor y' que, entre os pares armazenados localmente, minimiza a diferença absoluta entre y e y'. Desta forma podemos atribuir uma qualidade às respostas dos mineiros. Quanto maior for a diferença entre y e y' menor a qualidade do bloco. A pior resposta possível para cada desafio vai ser dada por 0 ou por 2^n. Este valor vai ser usado como referência para a qualidade da resposta dos utilizadores. Como existe a probabilidade de dois nós terem respostas com a mesma qualidade para o mesmo valor y, exigimos que em cada prova os mineiros invertam vários valores $y \in N$, que vai diminuir a probabilidade de um empate. O primeiro y é dado pelo hash do último bloco e os restantes são dados pelo hash do último valor y.

A regra de seleção de cadeia é um dos elementos mais importantes num protocolo de *blockchain*. Na nossa abordagem vamos usar uma variação da regra usada no Nakamoto Consensus [13]. Em vez de considerarmos que a cadeia ativa é a cadeia que precisou de mais poder computacional (a cadeia cuja soma das dificuldades das provas de trabalho é maior) para ser gerada, vamos considerar que a cadeia ativa é a cadeia cuja soma das qualidades das provas de espaço é maior e muito provavelmente foi a que precisou de mais espaço para ser construída.

A abordagem que acabamos de apresentar à primeira vista pode parecer correta. No entanto tem dois problemas. Ao usarmos como desafio o hash do último bloco estamos a dar ao mineiro controlo sobre o desafio do próximo bloco. Isto é, um mineiro malicioso consegue manipular o desafio do próximo bloco. Antes de adicionar um bloco à cadeia o mineiro pode ver se tem uma resposta favorável (qualidade elevada) ao desafio que é gerado pelo seu bloco. Se a sua resposta tiver baixa qualidade ele pode, por exemplo, mudar a ordem das transações do seu bloco. Esta alteração vai alterar o hash do bloco e vai gerar um desafio diferente para a prova de espaço O mineiro pode alterar o bloco tantas vezes quantas forem precisas até encontrar um desafio para o qual tem uma resposta com uma qualidade elevada. Disto, podemos concluir que a fonte de aleatoriedade dos desafios não pode ser a própria *blockchain*.

O segundo problema é que nada garante que há um intervalo de tempo entre os blocos. Se o tempo que um nó demora a criar um bloco for menor que o tempo de latência da rede vai ser mais difícil que as cadeias dos vários nós convirjam. Além disso, como estas provas podem ser construídas rapidamente, nada impede um nó malicioso de minerar uma grande quantidade de blocos de seguida. Mesmo que a qualidade das provas destes blocos não seja elevada, uma cadeia com um número suficiente de blocos de baixa qualidade pode ter um total de qualidade maior que uma cadeia com menos blocos de maior qualidade. Devido a estes problemas um nó malicioso com espaço limitado consegue rescrever a *blockchain* a partir do ponto que quiser.

Tendo em conta estes problemas decidimos criar uma lista com alguns requisitos que qualquer *blockchain* correta deve garantir: Requisito 1 - Impedir que um nó gere rapidamente uma cadeia contínua de blocos no topo da *blockchain*. Requisito 2 - Impedir que um nó consiga ganhar algum tipo de vantagem por gerar

blocos concorrentes para a mesma altura[6] da cadeia. Requisito 3 - Impedir que um nó produza um bloco de elevada qualidade para uma altura antiga, levando assim a uma reescrita da cadeia. Se a nossa abordagem ingénua for estendida de forma a garantir os seguintes requisitos, os seus problemas serão mitigados.

3.2 Etherspace

Nesta secção apresentamos a abordagem usada no Etherspace, que resolve todos os problemas identificados anteriormente.

Para resolver os problemas da abordagem ingénua, vamos começar por definir um oráculo, ao qual todos os nós têm acesso e cujo comportamento garante os requisitos apresentados na subsecção anterior. Este oráculo recebe como argumento um número que corresponde a uma altura na *blockchain* $h > 0$, e devolve o desafio de prova de espaço para o qual o bloco com a altura h tem de ter uma resposta. Este oráculo apenas vai devolver o desafio para a altura h, após um intervalo de x segundos, de ter devolvido o desafio para a altura $h - 1$. Inicialmente este oráculo apenas devolve o desafio para a altura 1, e para qualquer outra altura não vai devolver nada. Quando um nó pedir pela primeira vez o desafio da altura 1 o oráculo vai iniciar um temporizador de x segundos. No final desse temporizador o oráculo também vai passar a devolver o desafio da altura 2. Quando um utilizador pedir o desafio da altura 2 pela primeira vez, o oráculo vai iniciar outro temporizador de x segundos, e por aí em diante.

Ao utilizar este oráculo podemos garantir duas coisas: existe um intervalo de x segundos entre cada bloco e nenhum nó consegue influenciar o desafio do bloco seguinte. Com este oráculo e com as provas de espaço conseguimos garantir os requisitos que descrevemos. Nenhum nó vai conseguir gerar rapidamente uma cadeia de blocos no topo da *blockchain* pois o oráculo não revela o desafio do bloco seguinte até o temporizador expirar. O ritmo ao qual conseguimos adicionar blocos no topo da *blockchain* é limitado pelo temporizador do oráculo. Deste modo conseguimos garantir que existe um intervalo de x segundos entre cada bloco, garantindo o primeiro requisito. Como neste modelo o desafio do bloco seguinte é gerado pelo o oráculo e não pelo bloco que adicionamos à *blockchain*, os nós não conseguem influenciar o desafio do próximo bloco. Portanto mesmo que um nó gere múltiplos blocos para a mesma altura, não consegue obter qualquer tipo de vantagem, garantindo o segundo requisito. A regra de seleção de cadeia que usámos no modelo anterior em conjunto com o oráculo garante o terceiro requisito. Para conseguir provocar uma reescrita na cadeia o nó teria de conseguir produzir uma cadeia cuja qualidade fosse superior à ativa porque o número de blocos que um adversário consegue adicionar à cadeia é limitado pelo número de desafios que se consegue obter pelo oráculo. Para conseguir provocar a reescrita vai ser preciso possuir a maioria do espaço investido no protocolo. E assim garantimos o terceiro requisito.

[6] A altura de um bloco é o número de blocos que o precedem na cadeia. O primeiro bloco da cadeia tem altura zero. Um *fork* é o resultado de dois blocos que têm a mesma altura

Para implementar estes oráculos propomos duas possíveis soluções. Podemos usar um TEE como o SGX para implementá-los. Primeiro, podemos usar o TEE para gerar um temporizador (em todos os nós), que ao terminar devolve o desafio correspondente à prova de espaço que o próximo bloco na *blockchain* deve conter (este desafio será o mesmo para todos os nós da rede). Além disso, este desafio será sempre aleatório e poderíamos usar o TEE para verificar se este desafio é o correto. Uma vantagem que teria em relação ao PoET seria que um adversário não conseguiria comprometer o protocolo por corromper nós (este adversário poderia tentar que o seu TEE produzisse um desafio que lhe fosse mais favorável), qualquer nó correto iria perceber que tinha recebido uma mensagem de um nó malicioso.

A segunda forma é usar um comité, parecido ao utilizado no Byzcoin [22] para correr um protocolo de *coin flipping* [11], como o usado no Ouroboros [21], para gerar um desafio aleatório para o próximo bloco. Enquanto o protocolo de *blockchain* no qual este comité corre não estiver comprometido, podemos garantir que a maioria dos membros do comité são utilizadores honestos. Para conseguir manipular o desafio do próximo bloco o atacante ia precisar de controlar uma maioria dos elementos do comité. Como este protocolo vai precisar que os nós comuniquem entre si, a sua execução vai demorar algum tempo, que será limitado pela latência da rede. Logo conseguimos garantir que o intervalo de tempo entre cada bloco é o tempo que o protocolo de *coin flipping* demora a correr.

4 Avaliação

Nesta secção vamos avaliar o Etherspace, respondendo às seguintes perguntas: Como se compara o consumo energético do Etherspace com o do Ethereum? Qual é a frequência e o tamanho dos forks no Etherspace?

4.1 Metodologia

Nesta subsecção, vamos discutir os métodos que usámos para avaliar a nossa abordagem, tais como a implementação simplificada que usámos apenas para realizar testes.

Para realizar esta avaliação vamos usar como carga de trabalho transações reais retiradas do Etherscan [6] (do bloco 0x50f8f4 até ao bloco 0x5ef20).

Como termo de comparação para a nossa abordagem vamos usar uma versão alterada da versão 8 da implementação do Ethereum em Go. Nesta versão alterada, as provas de trabalho foram substituídas por um temporizador que segue uma distribuição de Poisson (o tempo entre cada bloco no Ethereum segue uma distribuição de Poisson). Optámos por correr os testes nesta versão alterada do Ethereum para poder correr múltiplos nós em apenas uma máquina.

Para testar a nossa abordagem, implementámos um protótipo do Etherspace. Este protótipo foi implementado em cima da versão 8 da implementação do Ethereum em Go. Numa versão inicial, para modelar as funções g e f optámos pelo método proposto em [8], pelo que utilizámos AES no modo CBC e truncamos o output (deste modo tornamos o AES numa função de um só sentido).

O bloco de 128 bits que damos a esta função é formado por zeros seguidos do *nonce* utilizado (que tem um tamanho fixo de n bits). Como vector de inicialização (IV), vamos utilizar o resultado de aplicar uma função de dispersão ao endereço do mineiro em conjunto com o *nonce* usado no bloco. No final de se computar o AES o seu output é truncado para n bits. Para o acesso aos dados necessários às provas de espaço ser mais rápido decidimos guardá-los numa btree, onde as chaves vão ser o contradomínio da função g_f que vão endereçar os pares de input (x, x') que as originam.

Para simplificar a avaliação, e podermos ter controlo sobre os valores produzidos pelo oráculo, implementámos o oráculo em Python. Este oráculo envia a todos os nós de Etherspace em intervalos predefinidos de tempo o desafio do próximo bloco. É de salientar que ambas as implementações do oráculo oferecem o mesmo comportamento (apenas o fazem de forma diferente), e que a correção do Etherspace apenas depende deste comportamento e não da implementação do oráculo. Deixámos as implementações concretas dos oráculos para trabalho futuro.

Para facilitar a realização dos testes com um número elevado de nós, decidimos substituir as provas de espaço do Etherspace por uma aproximação. Para descobrir qual é distribuição da qualidade das provas de espaço (as melhores provas de espaço têm uma qualidade mais alta) de vários nós, corremos uma simulação na qual enviámos desafios de proof-of-space a 500 nós e recolhemos as suas respostas. Para facilitar este processo usámos tamanhos para as provas de espaço baixos. O tamanho de cada valor na tabela da função g_f (nomeadamente x, x', y) é cada um 16 bits, pelo que a tabela ideal de g_f teria 65 536 entradas. Na prática, as tabelas de cada utilizador são diferentes (são usados alguns dados específicos tais como o endereço da conta) e alguns dos valores y não têm uma inversão em algumas tabelas. As tabelas que podem ser criadas por uma conta de Ethereum (com estes parâmetros para as provas de espaço) vão ter no máximo perto de 24 000 entradas. Cada um dos nós que usámos para extrapolar a distribuição dos dados tinha uma tabela com o número máximo de entradas possível. Cada desafio obrigava os nós a inverter 100 valores, onde cada valor y que o nó tinha de inverter era dado por efectuar duas vezes a função de dispersão Keccak-256. Recolhemos as respostas destes nós a aproximadamente 42 000 desafios de provas de espaço. Verificámos que a distribuição da qualidade das respostas dos nós seguia uma distribuição normal com uma média de 95 514.2 e desvio padrão de 470.7. Para efectuar esta avaliação em vez de termos os nós a criar provas de espaço (que vão ter uma certa qualidade), usámos um gerador de números aleatórios que atribui a cada nó o valor da qualidade que a sua prova de espaço tem para cada altura da cadeia. Esta simplificação permite-nos correr mais testes num curto espaço de tempo, sem afetar as conclusões gerais obtidas.

Para avaliar a nossa solução usámos um computar com um CPU Intel Xeon Gold 6138 com 20 cores e frequência de $2GHz$ e $65GB$ de RAM. Corremos as duas versões do Ethereum durante uma hora com um total de 60 nós honestos, dos quais todos estão a minerar, e sem nós maliciosos.

4.2 Consumo de Energia

Para comparar o consumo de energia do Etherspace com o do Ethereum vamos usar algumas estimativas. O número estimado de nós no Ethereum na altura da escrita deste artigo é de 8 311 [5]. Se o Etherspace tivesse o mesmo número de nós e cada nó tentasse responder a todas as provas de espaço, cada nó realizaria um total de 198 vezes a função de dispersão (se usarmos o mesmo número de iterações que usámos para retirar a distribuição das qualidades das provas na subsecção anterior, cada nó vai ter de gerar 99 valores adicionais para inverter a partir do primeiro valor, para gerar cada um destes valores vai ter de realizar duas chamadas à função de dispersão) para gerar todos os valores que tem de inverter em cada prova. No total seriam 1 645 578 chamadas à função de dispersão por bloco. Na altura da escrita deste artigo a dificuldade das provas de trabalho é de aproximadamente $2\,035TH$ [6], ou seja, que conseguimos encontrar uma solução para uma prova de trabalho uma vez para cada $2\,035T$ chamadas à função dispersão. Podemos usar este valor como estimativa para o número de funções de dispersão que a rede de Ethereum precisa por bloco (que na realidade provavelmente vai ser um pouco maior). O número de funções de dispersão que são precisas para adicionar um bloco à *blockchain* do Ethereum é maior por várias ordens de magnitude ao número do Etherspace. Como em ambos os sistemas é usada a mesma função de dispersão, podemos concluir que o consumo energético do Etherspace é muito menor que o do Ethereum.

4.3 Convergência da blockchain

Para analisar a convergência da *blockchain*, contámos o número de vezes em que os nós abandonam a cadeia que estavam a minerar, a ocorrência de *forks*. Na nossa abordagem todos os nós criam blocos para todas as alturas, pelo que há muitos *forks* de um bloco, o mais recente, que é abandonado quando o mineiro encontra um bloco de melhor qualidade para a mesma altura. Portanto, para este teste apenas tivemos em conta *forks* cujo tamanho é superior a um bloco.

Após correr ambos os protocolos durante uma hora, a *blockchain* do Ethereum tinha 210 blocos, e a *blockchain* do Etherspace tinha 174 blocos. Os resultados que obtivemos estão nas figuras 1 e 2. Como podemos observar a frequência e o tamanho dos forks no Etherspace é menor que no Ethereum.

5 Conclusão

As provas de trabalho, apesar de eficazes em manter a segurança dos protocolos de *blockchain*, desperdiçam uma grande quantidade de energia. Este trabalho explorou as condições necessárias e suficientes para utilizar provas de espaço numa *blockchain*. Uma *blockchain* que use provas de espaço vai ter um consumo de energia muito menor, vai oferecer garantias de segurança semelhantes e para além disso a *blockchain* vai tornar-se mais descentralizada. Isto deve-se a ser provável que utilizadores com menos recursos se juntem ao protocolo, pois o custo energético de correr um nó de etherspace é baixo e o custo de memória é menor que o custo de hardware específico para computar provas de trabalho.

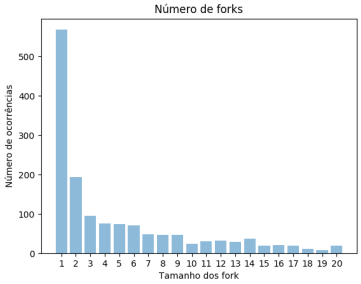

Fig. 1. Frequência e tamanho de forks no Ethereum

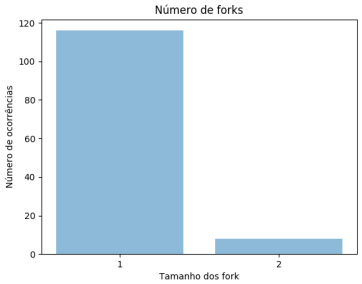

Fig. 2. Frequência e tamanho de forks no Etherspace

As provas de espaço, embora promissoras, ainda não foram concretizadas em nenhum sistema prático. Este artigo propõe e avalia uma solução nesta direcção.

5.1 Trabalho Futuro

Este artigo foca-se nas condições necessárias para substituir as provas de trabalho pelas provas de espaço numa *blockchain*. Futuramente vamos implementar e testar as duas implementações de oráculos que propusemos na Secção 3.2. Além disso, também vamos testar a resistência deste protocolo a utilizadores maliciosos e implementar alguns ataques.

6 Agradecimentos

Este trabalho é parcialmente financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER) através do Programa Operacional Regional de Lisboa e por Fundos Nacionais através da FCT - Fundação para a Ciência e a Tecnologia no âmbito do projeto Lisboa-01-0145-FEDER-031456 (Angainor) e UID/CEC/50021/2019.

References

1. Bitcoin energy consumption index. https://digiconomist.net/bitcoin-energy-consumption, accessed: 17-11-2018
2. Burstcoin. https://www.burst-coin.org/, accessed on 8-06-2019
3. Chia network. https://chia.net, accessed: 01-06-2019
4. Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper, accessed: 01-06-2019
5. Ethernodes. https://ethernodes.org/network/1, accessed: 11-06-2019
6. Etherscan - ethereum blockchain explorer. https://etherscan.io/, accessed: 11-06-2019
7. Hyperledger sawtooth. https://sawtooth.hyperledger.org, accessed: 31-05-2019

8. Abusalah, H., Alwen, J., Cohen, B., Khilko, D., Pietrzak, K., Reyzin, L.: Beyond hellman's time-memory trade-offs with applications to proofs of space. Cryptology ePrint Archive, Report 2017/893 (2017)
9. Aspnes, J., Jackson, C., Krishnamurthy, A.: Exposing computationally-challenged Byzantine impostors. Tech. Rep. YALEU/DCS/TR-1332, Yale University Department of Computer Science (Jul 2005)
10. Back, A.: Hashcash - a denial of service counter-measure. Tech. rep. (2002)
11. Blum, M.: Coin flipping by telephone. pp. 11–15 (01 1981)
12. Boneh, D., Bünz, B., Fisch, B.: A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712 (2018)
13. Bonneau, J., Miller, A., Clark, J., Narayanan, A., A. Kroll, J., W. Felten, E.: Sok: Research perspectives and challenges for bitcoin and cryptocurrencies (07 2015)
14. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: On security analysis of proof-of-elapsed-time (poet). In: Spirakis, P., Tsigas, P. (eds.) Stabilization, Safety, and Security of Distributed Systems. Springer International Publishing, Cham (2017)
15. Cohen, B.: Stanford seminar - stopping grinding attacks in proofs of space. https://www.youtube.com/watch?v=2Zlcgt8FVz4, accessed: 01-06-2018
16. Douceur, J.R.: The sybil attack. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. IPTPS '01, Springer-Verlag, London, UK (2002)
17. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology. CRYPTO '92, Springer-Verlag, London, UK (1993)
18. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. Cryptology ePrint Archive, Report 2013/796 (2013)
19. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. SOSP '17, ACM, New York, USA (2017)
20. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019 (2015)
21. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889 (2016)
22. Kogias, E.K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. In: 25th USENIX Security Symposium (USENIX Security 16). Austin, TX (2016)
23. Malone, D., O'Dwyer, K.: Bitcoin mining and its energy footprint (01 2014)
24. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf (2008)
25. Park, S., Kwon, A., Fuchsbauer, G., Gaži, P., Alwen, J., Pietrzak, K.: Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528 (2015)
26. Pietrzak, K.: Simple verifiable delay functions. Cryptology ePrint Archive, Report 2018/627 (2018)
27. Vukolić, M.: The quest for scalable blockchain fabric: Proof-of-work vs. bft replication (05 2016)
28. Wesolowski, B.: Efficient verifiable delay functions. Cryptology ePrint Archive, Report 2018/623 (2018)
29. "QuantumMechanic", U.: Proof of stake instead of proof of work. https://bitcointalk.org/index.php?topic=27787.0, accessed: 31-05-2019