
# Loteria Descentralizada em Blockchain EOSIO

**Ricardo de Barros Marlière**

JUIZ DE FORA

DEZEMBRO, 2018

# Loteria Descentralizada em Blockchain EOSIO

Ricardo de Barros Marlière

Universidade Federal de Juiz de Fora

Intituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Sistemas de Informação

Orientador: Alex Borges Vieira

JUIZ DE FORA

DEZEMBRO, 2018

# Loteria Descentralizada em Blockchain EOSIO

Ricardo de Barros Marlière

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Alex Borges Vieira
Doutor em Ciência da Computação

Heloísa Pinna Bernardo
Doutora em Contabilidade e Controladoria

Mário Antônio Ribeiro Dantas
Doutor em Ciência da Computação

JUIZ DE FORA

29 DE DEZEMBRO, 2018

# Resumo

Em 2008, um pesquisador anônimo publicou sua mais nova invenção: o Bitcoin, protocolo ponto a ponto de dinheiro eletrônico descentralizado que está rapidamente causando mudanças significativas no setor financeiro. Menos de 10 anos depois, sua tração continua forte atraindo cada vez mais investidores e especuladores, inaugurando uma nova subárea na Ciência da Computação que é o estudo das *blockchains*. Numerosos são os debates acerca da escalabilidade do sistema proposto por Nakamoto (2008) e, neste sentido, o foco deste trabalho é explorar de forma prática uma nova arquitetura escalável proposta por Larimer (2017b), chamada EOSIO.

**Palavras-chave:** Bitcoin, criptomoeda, descentralização, *blockchain*, EOSIO

# Abstract

In 2008, an anonymous researcher published his newest invention: the Bitcoin, a peer to peer decentralized protocol of electronic cash that is quickly and significantly disrupting the finantial sector. Less than 10 years later, its traction keep steadily growing and attracting more and more investors and speculators, inaugurating a new subarea of Computer Science which is the study of blockchains. There are great and extensive discussions about the scalability of the system proposed by Nakamoto (2008) and, in that sense, the goal of this work is to explore a new scalable architecture called EOSIO, proposed by Larimer (2017b), in a practical manner.

**Keywords:** Bitcoin, *cryptocurrency*, *decentralization*, *blockchain*, EOSIO

*"The joy of life consists in the exercise of one's energies, continual growth, constant change, the enjoyment of every new experience. To stop means simply to die. The eternal mistake of mankind is to set up an attainable ideal".*

*Aleister Crowley*

# Conteúdo

# Lista de Figuras

# Lista de Abreviações

| | |
|---|---|
| ABI | Application Binary Interface |
| API | Application Programming Interface |
| BIP | Bitcoin Improvement Proposal |
| BTC | Bitcoin |
| BP | Block Producer |
| CAPTCHA | Completely Automated Public Turing test to tell Computers and Humans Apart |
| CDT | Contract Development Toolkit |
| CLI | Command Line Interface |
| DAC | Decentralized Autonomous Corporation |
| DAO | Decentralized Autonomous Organization |
| Dapp | Decentralized Application |
| DPoS | Delegated Proof of Stake |
| ECAF | EOS Core Arbitration Forum |
| ETC | Ethereum Classic |
| ETH | Ethereum |
| EVM | Ethereum Virtual Machine |
| JSON | JavaScript Object Notation |
| LCG | Linear Congruential Generator |
| PRNG | Pseudorandom Number Generator |
| PoS | Proof of Stake |
| PoW | Proof of Work |
| RPC | Remote Procedure Call |
| SDK | Software Development Kit |
| UTXO | Unspent Transaction Output |

# 1 Introdução

A crescente demanda por criptomoedas por um público cada vez mais abrangente é demonstrada através do aumento significativo da capitalização do mercado ao longo dos últimos anos. Ao final de 2017, estima-se que o valor de mercado conjunto de todas as criptomoedas fosse superior à 800 bilhões de dólares estadunidenses, sendo que mais de 45% correspondiam apenas ao mercado de bitcoins em si[1]. Isto representa uma diferença muito acentuada para com o ano de 2016, onde o montante não excedeu muito além de 16 bilhões.

Tal demanda fica justificada pelo fato da nova tecnologia inerente a estas moedas, chamada *blockchain* (cadeia de blocos, em português), possibilitar transações seguras de valor de forma direta entre as pessoas, sem uma instituição financeira como intermediária. Em momentos de crises econômicas generalizadas, é prática comum recorrer a metais preciosos para estabelecer o que é conhecido como reserva de valor e uma nova alternativa surge para tal com o advento das criptomoedas. Prova disto é o aumento da procura por bitcoins em países afetados por grave inflação monetária como Venezuela e Zimbábue, o que resulta numa cotação interna muito superior ao restante dos mercados do mundo.

Por outro lado, por se tratar de um protocolo de rede ponto a ponto, criptomoedas podem ser utilizadas com motivações eticamente duvidosas. Ao contrário do sistema bancário, onde cada conta é sempre vinculada à identidade de alguém, um endereço sob controle de um nó da rede nem sempre pode ter sua pessoa rastreada. Ao aliar o novo dinheiro eletrônico que possui alto grau de anonimidade com um outro protocolo chamado Tor que obfusca o roteamento de pacotes para dificultar seu rastreio, Ross Ulbricht fundou em Fevereiro de 2011 uma das primeiras comunidades onde o Bitcoin foi adotado. O mercado negro chamado Silk Road, onde criminosos negociavam artigos ilícitos usando a moeda, foi em parte grande responsável pela popularização do tema e teve suas operações completamente desativadas pelo FBI apenas em Outubro de 2013, com seu fundador condenado a cumprir pena vitalícia em regime fechado.

[1]Disponível em: ⟨https://coinmarketcap.com/⟩ (acesso em 17-11-2018)

Tendo estabelecido o marco inicial em 2008 pela invenção da *blockchain*, o *cypherpunk* (HUGHES, 1993) Satoshi Nakamoto foi também responsável pela primeira implementação do protocolo e permaneceu ativo como desenvolvedor até 2010. Desde então, muitas evoluções vêm tomando forma tanto no contexto do próprio Bitcoin quanto em outros protocolos. Os primeiros projetos de *blockchains* independentes do Bitcoin que buscavam inovar e refinar o conceito datam de 2012 mas ainda há muito o que ser explorado haja vista que são escassas as evidências de casos de uso atualmente implementados que possuam alguma notável adoção pelo público em geral que não seja uma simples moeda.

Como é de conhecimento comum, a maior de todas as empresas que oferecem serviços de transporte privado não possui frota alguma; de maneira semelhante, a empresa dona das mídias da maior rede social do mundo não produz conteúdo; o maior varejista não possui estoque; o maior provedor de acomodações não possui imóveis. A adoção em massa é um grande indicativo do sucesso de uma aplicação e os gigantes corporativos do setor da tecnologia demonstram que existe demanda para diversos casos de uso onde o valor é derivado diretamente da comunidade. Comércio virtual, redes sociais, cassinos, cartórios, eleições, cadeias de custódia, cadeias produtivas como por exemplo no rastreio e certificação da origem de madeira ou diamantes, casas de câmbio e bolsas (ou *exchanges*, no inglês) e muitas outras aplicações podem ser modeladas como casos de uso descentralizados e passam a representar uma grave afronta aos modelos tradicionais por serem mais ou tão seguros quanto, transparentes, autômatos e potencialmente menos custosos.

## 1.1 Problema

O problema resolvido pelas *blockchains* é a remoção de intermediários em transações de valor. Indo além do Bitcoin e instituições financeiras, podem ser modelados muitos outros casos de uso amplamente adotados na atualidade através da criação de contrapartes descentralizadas. Para o âmbito deste trabalho, é proposta uma *dapp* de loteria.

## 1.2 Justificativa

Utiliza-se o novo paradigma de negócio possibilitado pelo conceito da *blockchain* para remover o intermediário no contexto desta aplicação, pois a falta de transparência compromete a verificação da satisfatoriedade estatística de diversos jogos.

## 1.3 Objetivo

O sucesso de determinada tecnologia ou aplicação pode ser intuitivamente relacionado com o tamanho de sua base de usuários. Apesar da *blockchain* ser uma invenção recente, fato é que sua adoção é cada vez maior e a tendência é permanecer em crescimento desde que mais casos de uso sejam implementados de forma descentralizada. Para tanto, o objetivo da plataforma proposta neste trabalho é a criação de uma comunidade de jogadores visando o aumento da base de usuários de soluções descentralizadas como um todo.

## 1.4 Metodologia

Para tornar viável o desenvolvimento deste trabalho, um mapeamento da literatura é necessário para estabelecer a base teórica e devida contextualização do tema via exemplos. Segue-se então para a especificação dos requisitos da aplicação que são por sua vez utilizados para modelar a arquitetura da mesma.

# 2 Fundamentação Teórica

O objetivo do presente capítulo é apresentar um histórico e panorama geral das principais tecnologias presentes nas criptomoedas. Na seção 2.1, aborda-se o protocolo pioneiro de dinheiro eletrônico Bitcoin. Na seção 2.2 o inovador Ethereum é visto em detalhes e por último, na seção 2.3, é feita uma análise das possibilidades apresentadas pelo projeto EOSIO.

## 2.1 Bitcoin

Em 2008, um pesquisador conhecido apenas pelo pseudônimo Satoshi Nakamoto publicou um artigo simples de apenas oito páginas em que apresenta um sistema ponto a ponto de dinheiro eletrônico que possui implicações sociais e econômicas ainda a serem apreciadas em sua plenitude. O Bitcoin torna possível a transferência direta de valor entre dois pontos ao redor do mundo de forma segura, rápida e barata. Na prática, o que ocorre é a remoção do atravessador (como, por exemplo, o próprio banco no contexto de transações bancárias), devido ao fato de ser uma rede descentralizada (NAKAMOTO, 2008). Respostas diversas são extraídas a depender de quem seja perguntado: Bitcoin poderia ser considerado, ao mesmo tempo, dinheiro programável, uma nova classe de ativos, uma moeda, uma rede de computadores e uma commodity.

Note que, apesar de ser pioneiro na efetiva resolução deste problema vinculado à moedas centralizadas, o Bitcoin não foi a primeira tentativa. De fato, a primeira proposta de criação de dinheiro eletrônico, eCash (CHAUM, 1982), requeria dos usuários que depositassem confiança numa autoridade central, muito similar ao sistema bancário. Isso só foi resolvido meio quarto de século mais tarde por Satoshi Nakamoto ao criar um protocolo que tomou emprestado conceitos importantes oriundos de: Pretty Good Privacy (ZIMMERMANN, 1995), b-money (DAI, 1998), Proof of Work (PoW) (JAKOBSSON; JUELS, 1999), Hashcash (BACK, 2002), BitTorrent (COHEN, 2003), Reusable Proofs of Work (FINNEY, 2004), Bit Gold (SZABO, 2005).

Parte-se da ideia de que, desde que dinheiro nada mais é que um sistema contábil, o livro razão (*ledger*) deve ser fielmente distribuído a todos os pares membros da rede. Porém, ao contrário de um sistema centralizado em que o banco detecta fraudes em que se tenta gastar mais dinheiro do que se tem disponível (no inglês, *double-spending*), o mesmo não é tão simples num sistema distribuído. Afinal, cópias digitais são triviais de serem feitas e é necessario um mecanismo de armazenamento em que a situação seja improvável. Este problema foi resolvido com o Bitcoin. Para tal, Nakamoto (2008) propôs "uma rede ponto-a-ponto que usa PoW para registrar um histórico público de transações que rapidamente se torna computacionalmente impraticável para um atacante o alterar se nós honestos controlarem a maior parte do poder de CPU".

Dentre as principais características do protocolo, pode-se citar em destaque as três maiores que evidenciam a verdadeira quebra de paradigmas trazida à tona pelo Bitcoin:

1. Existe um limite imposto pelo *software* de forma que o total de bitcoins em circulação jamais será superior a 21 milhões, o que a torna deflacionária partindo da hipótese de demanda crescente.

2. A impressão de novas moedas não depende da autoridade de um Banco Central, o que a torna livre de quaisquer fronteiras[2].

3. Pela primeira vez na história da humanidade, pessoas comuns não precisam recorrer a bancos para armazenamento e transferência seguras de valor[3].

As subseções a seguir visam elucidar os componentes e conceitos mais importantes do protocolo para que o mesmo possa ser mais facilmente entendido. O mais importante é a transação criptograficamente válida de moedas e será visto na subseção 2.1.1. Tais transações devem ser agrupadas em blocos e enumeradas de acordo com o *timestamp* em que foram geradas, sendo os detalhes discutidos na subseção 2.1.2. Por último, a subseção 2.1.3 explica o mecanismo que incentiva os nós da rede a permanecerem honestos.

[2]Entretanto, derivativos (contratos futuros) servem como ferramenta para autoridades e grandes instituições financeiras controlarem o preço da moeda, tal como é feito nos mercados de metais preciosos e outras *commodities* (ANGEL; MCCABE, 2008).

[3]Chaves privadas ocupam o mesmo espaço em disco independentemente da quantidade disponível de bitcoins numa determinada carteira (NAKAMOTO, 2008).

### 2.1.1 Transação

Uma transação nada mais é que o envio de moedas de um endereço a outro, anunciada publicamente na rede. Nakamoto (2008) definiu uma moeda eletrônica como uma cadeia de assinaturas digitais. Quando ocorre a transferência de moedas, o dono assina o *hash* da transação anterior que lhe proveu a propriedade de suas moedas em primeiro lugar junto à chave pública do próximo dono. Este, por sua vez, consegue verificar a procedência de suas novas moedas através da verificação criptográfica deste encadeamento formado (ilustrado na figura 2.1). A função criptográfica em uso pelo Bitcoin é a SHA-256 (EASTLAKE; HANSEN, 2011). O problema, então, reside no fato de que quem está recebendo o pagamento não saberia dizer se um dos donos anteriores fraudou o sistema via *double-spending*, a não ser que todos os participantes da rede chegassem a um consenso acerca do histórico único de transações, a *blockchain*.

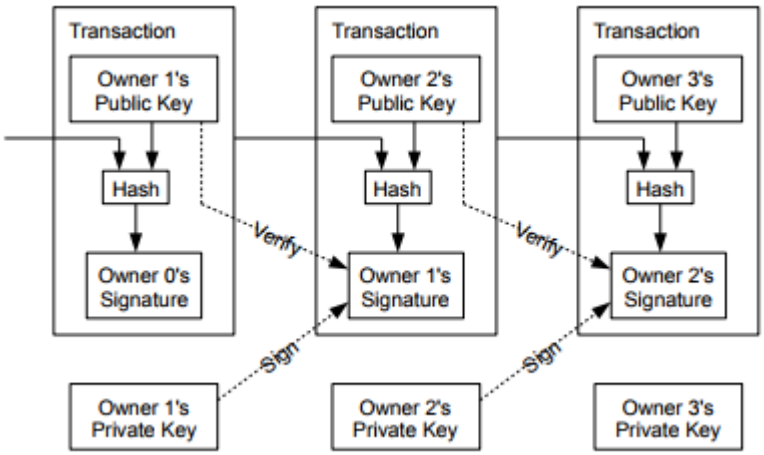


Figura 2.1: Modelo de transações com encadeamento de propriedade

De fato, todas as bitcoins podem ter seus traços perseguidos ao longo de todas as transações em que estiveram envolvidas, visto que todos os nós da rede operam com uma cópia fiel do mesmo livro razão. Ao final, sempre encontra-se o bloco em que tenham sido mineradas. Cada moeda que ainda não foi gasta tem sua origem na saída de uma transação - UTXO.

### 2.1.2 Cadeia de Blocos

A solução proposta por Nakamoto (2008) requer a utilização de um sistema distribuído de *timestamp*, pois assim prova-se que os dados existiram num determinado momento. Desta forma, cada transação ocorre numa data e hora específica e assim que for publicada na rede ficará em estado pendente. Para ser confirmada, a transação deve ser incluída num bloco válido encontrado por um nó mineiro. Um bloco é uma estrutura de dados que possui como informação principal um conjunto de transações. Para que o sistema de *timestamp* funcione, cada bloco possui o *hash* do bloco anterior, formando uma lista encadeada. A figura 2.2 ilustra o processo.

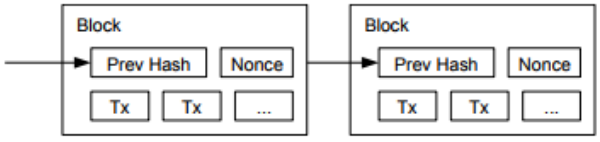


Figura 2.2: Modelo da blockchain

O processo de descoberta de novos blocos, também chamado de mineração e responsável por garantir consenso entre todos os nós, ocorre através de um algoritmo de força bruta como PoW. Tal algoritmo tenta exaustivamente encontrar um valor (*nonce* - incluído na estrutura do bloco) até que o *hash* do bloco comece com um número específico de bits zero, definido dinamicamente pela rede. Quanto mais bits requeridos, mais difícil se torna a mineração e, com isso, a rede ajusta-se naturalmente ao aumento ou diminuição do total de processamento (medido em *hashrate*) fornecido pelos mineiros. Em média, a dificuldade é exponencial à quantidade de bits requisitados. Assim, a alteração de um bloco já inserido na rede necessariamente implica na realização deste trabalho novamente, inclusive para os blocos que o sucedem, visto que o *hash* de cada bloco é incluído em seu sucessor, reforçando a segurança da rede a cada iteração (NAKAMOTO, 2008).

Para economizar espaço de disco e agilizar o processo de sincronização com a rede, o bloco inclui a raiz da árvore de assinaturas (MERKLE, 1980) de suas transações. Isto possibilita a verificação de pagamentos de forma confiável (desde que nós honestos sejam maioria da rede) sem despender processamento de verificação individual de cada transação, pois a raiz da árvore - Merkle Root, um *hash* - está integrada ao *hash* do bloco.

### 2.1.3 Mineração

Por convenção, a primeira transação de um novo bloco coloca em circulação novas bitcoins, em posse daquele que o encontrou. Trata-se de uma recompensa[4] para que os nós continuem a prover a segurança necessária da rede de forma honesta. Porém, o montante cai pela metade em intervalos de 210 mil blocos, até que todas as bitcoins sejam mineradas até o limite de 21 milhões[5] (NAKAMOTO, 2008).

Um nó mineiro tem duas responsabilidades: validar o conteúdo das transações que aguardam serem incluídas num bloco e rodar o processo de mineração que é a única maneira de se descobrir novos blocos. As transações que ainda não foram verificadas e inseridas num bloco ficam pendentes numa fila e podem receber prioridade pelos mineiros de acordo com a quantidade despendida em taxas. As taxas de transação serão a única forma de receita dos mineiros quando todas as bitcoins já estiverem em circulação.

É importante notar que a verificação criptográfica do conteúdo da *blockchain* não é tão computacionalmente exigente quanto a mineração. O funcionamento do algoritmo que é utilizado como PoW no Bitcoin para mineração, este sim extremamente dispendioso, torna aleatória a escolha do mineiro a descobrir o próximo bloco da cadeia. Quanto maior o *hashrate* disponibilizado por um mineiro, maior a probabilidade dele encontrar um bloco válido a ser aceito pela rede. A segurança dos dados da *blockchain* é diretamente proporcional à quantidade total de *hashrate* na rede. A competição por maiores fatias do *hashrate* total provido para a rede, que visa o acúmulo da moeda emitida em cada bloco, é importante para garantir a integridade e honestidade de pelo menos 51% deste, sendo a solução encontrada por Nakamoto (2008) para resolver o problema dos generais bizantinos nesta rede (LAMPORT; SHOSTAK; PEASE, 1982).

[4]Ou custo, a depender da perspectiva. Na emissão de novas moedas a cada bloco, existe uma taxa de inflação matematicamente previsível que independe do volume de *hashrate*. Desta forma, a comunidade coletivamente custeia um salário pelo serviço de mineiro.

[5]Vale ressaltar que existem muitas bitcoins perdidas, pois uma vez que se perde o controle de sua chave privada, você fica impossibilitado de utilizá-las. Além disso, estima-se que um grande montante de bitcoins originados nos primeiros meses da rede, sob controle de Satoshi Nakamoto, não serão jamais utilizados. Portanto, na prática, o montante em circulação será inferior ao limite.

## 2.2 Ethereum

A chamada segunda geração de criptomoedas surgiu com a proposta de utilizar-se da segurança provida por uma *blockchain* na construção de protocolos mais complexos, servindo como uma camada fundamental (TSCHORSCH; SCHEUERMANN, 2015). Para tal, é necessário ir além do sistema de transações do Bitcoin que oferece recursos limitados em seus *scripts* de transações, que basicamente descrevem regras para que o receptor consiga ganhar acesso às moedas.

Para possibilitar casos de usos mais elaborados obtidos ao explorar-se conceitos como o de contrato inteligente (SZABO, 1997), Ethereum é um protocolo que objetiva fornecer aos desenvolvedores uma linguagem de programação Turing-completa (BUTERIN, 2014). A falta de artifícios como *loops* na linguagem de *scripting* do Bitcoin impossibilita a escrita de muitos tipos de contratos[6], assim como a simplicidade do estado binário atribuído a uma UTXO - despendida ou não (LARIMER, 2017a).

Uma *blockchain* pode ser vista como sendo um sistema de transições. O estado atual é computado através de uma função que recebe como entrada um estado e uma transação e gera como saída um novo estado. Tal função pode ser deliberadamente escrita na forma de um contrato em alguma das linguagens de alto nível suportadas como a Solidity, sendo o código resultante executado na EVM. O custo computacional para executar o código de um contrato pelos mineiros é pago pelo usuário e deduzido do número de iterações necessárias na EVM sendo incluso na transação os campos *startgas* (o número máximo de iterações) e *gasprice* (o valor despendido pelo usuário para cada uma). Desta forma evita-se ataques de *spam* como no caso de um *loop* infinito no código de um contrato (BUTERIN, 2016).

Ao contrário do Bitcoin, a inflação no protocolo Ethereum não possui um limite superior. Os mineiros continuarão colocando cada vez mais moedas em circulação (LUBIN, 2014). Preocupados com a escalabilidade do protocolo, os desenvolvedores da Ethereum Foundation enveredaram por uma linha de pesquisa onde pretendem atualizar

[6]Entretanto, trata-se de um experimento em andamento. A proposta de melhoria do Bitcoin (BIP) 141, adotada em consenso e conhecida como SegWit (CORE, 2016), faz com que o versionamento seja levado em conta num *script*. Isto remove a necessidade de um *hardfork* para novas atualizações da linguagem.

o mecanismo de consenso para PoS (BUTERIN; GRIFFITH, 2017). Também pretendem adotar os conceitos da Lightning Network do ecossistema Bitcoin via Raiden Network, onde a ideia fundamental é escalar o sistema para camadas *off-chain* de computação de forma a minimizar o uso da *blockchain* que torna-se apenas a camada fundamental para assentamento do estado das operações realizadas nos chamados *state channels* (POON; BUTERIN, 2017).

### 2.2.1 Contrato Inteligente

Szabo (1997) sugere que muitos tipos de contratos podem ser embutidos em *hardware* e *software*. O exemplo mais simples é a máquina de vendas, onde o interessado insere dinheiro na entrada da mesma que fica a cargo da gestão de seus recursos disponíveis. Para tanto, as regras contidas em seu programa equivalem a cláusulas contratuais.

Além de moedas, a aplicabilidade é diversa. Por exemplo, contratos podem ser utilizados para a criação de *dapps* como derivativos e outros títulos financeiros, sistemas de reputação e identidade, armazenamento de arquivos, conta-poupança, testamento, etc (BUTERIN, 2014). Pode-se, inclusive, botar em prática o conceito de Empresa Autômata Descentralizada (DAC) (LARIMER, 2013)[7]. Abaixo um exemplo de o que talvez seja o contrato mais simples possível de se implementar em Solidity para Ethereum:

```
1 pragma solidity ^0.4.22;
2 contract hello_world {
3  function render () public pure returns (string) {
4    return 'hello world';
5  }
6 }
```

Em qualquer negócio existem, basicamente, dois grupos de indivíduos: aqueles que produzem valor e aqueles que o consomem. Porém, é muito comum, no contexto do paradigma centralizado, que as partes envolvidas possuam interesses desalinhados por serem intermediadas pelo dono do negócio. O acionista zela pelo lucro, pela saúde financeira da empresa e tem particular interesse em relatórios e balanços periódicos, etc. O

[7]No dia 25 de Julho de 2017, a SEC dos Estados Unidos emite uma nota dizendo que determinado conjunto de leis estadunidenses poderia ser extendido à organizações virtuais. Um ano antes, uma falha foi explorada no contrato inteligente TheDAO na rede Ethereum que permitiu o desvio das moedas acumuladas pelo mesmo, gerando prejuízo avaliado em mais de 100 milhões de dólares e um *hardfork* na rede.

produtor também visa a maximização de seu lucro mas depende diretamente do tempo que dispõe para obtê-lo. Em contrapartida, o usuário não se importa com o negócio em si e apenas espera uma plataforma de fácil uso, eficiente e barata.

Ao modelar um negócio de forma descentralizada através de contratos inteligentes, os interesses de todas as partes se alinham. Afinal, a própria empresa e todo seu capital pode ser representada pelos *tokens* criados, criando assim um laço econômico entre o produtor e o usuário. Além de não precisarem mais pagar uma ou várias taxas de intermediação (definidas pelo acionista) por não serem os donos dos meios de produção, os usuários são incluídos no modelo fechando, assim, um ciclo econômico eficiente onde todos são como *shareholders* - ou *tokenhodlers*.

## 2.3 EOSIO

Um dos maiores debates em pauta nas comunidades das maiores *blockchains* que adotam PoW como mecanismo de consenso gira em torno da escalabilidade do sistema. Originalmente, Nakamoto (2008) propôs uma solução linear temporária que é o aumento da capacidade de cada bloco via um *hardfork*. Entretanto, novos mecanismos ganharam destaque, principalmente após a segunda geração de *blockchains*.

Surgiu em 2012 a Peercoin, com a ideia de construir uma *blockchain* segura sem que houvesse enormes gastos com energia elétrica como acontece na mineração em redes PoW[8] - como visto no comparativo da figura 2.3. O mecanismo de consenso PoS foi proposto, embora na prática os desenvolvedores optaram por implementar um mecanismo híbrido PoW/PoS. As dificuldades que enfrentaram na tentativa de usar apenas PoS são explicadas por diversos críticos do mecanismo, como Poelstra (2015).

Existem diversas variações do PoS, sendo que a diferença principal esteja na forma de selecionar o próximo nó a validar e inserir um novo bloco. Em sua forma mais simples, quanto maior o *stake*, isto é, o montante de *tokens* em posse de um nó, maiores as chances dele ser escolhido para validar o próximo bloco (KING; NADAL, 2012). Pode-se

[8]Estima-se que, anualmente, os mineiros em conjunto gastem bilhões de dólares com custos de energia elétrica. Trata-se de uma inflação velada.

[9]Mineiros em conjunto, comparados com o consumo individual de eletricidade de outros países. Disponível em: ⟨https://digiconomist.net/ethereum-energy-consumption⟩ (acesso em 17/11/2018).

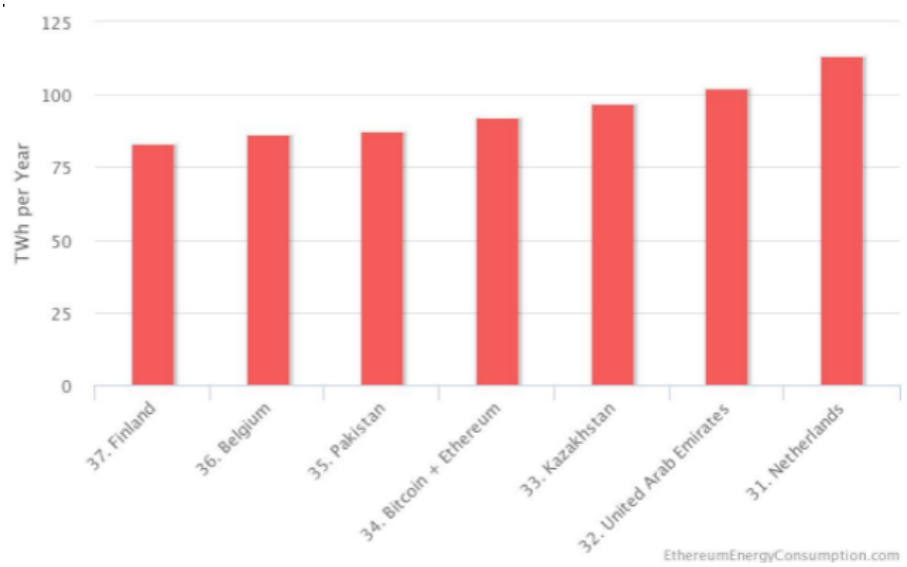


Figura 2.3: Consumo das redes BTC e ETH[9]

traçar uma analogia com o sistema lotérico onde cada moeda corresponde a um bilhete e, portanto, quanto mais bilhetes maior a probabilidade de acerto[10].

Indo mais além, o mecanismo DPoS parte do princípio de que os BPs (análogos aos mineiros) devem ser eleitos[11]. Desta forma, um nó ganha ou perde função de validador através do voto da comunidade (LARIMER, 2014). Isto possibilita muitas otimizações, visto que enquanto o *throughput* de transações por segundo suportado por redes PoW esteja limitado a poucas dezenas na melhor hipótese e o próximo bloco a ser minerado não possui *timestamp* previsível (podendo chegar a horas de intervalo, a depender da estabilidade da rede), *blockchains* que adotam DPoS possuem capacidade para milhares com intervalo entre blocos fixado (SCHUH; LARIMER, 2015), sendo de apenas meio segundo em EOSIO.

Após o êxito no *bootstrap* de ambos os projetos Bitshares e Steemit, respectivamente uma *exchange* e uma rede social descentralizadas, Larimer (2017b) divulga em Maio de 2017 seu novo projeto: EOSIO. A proposta de uma nova arquitetura de *blockchain* idealiza atingir escalabilidade horizontal e vertical para *dapps*. Diferentemente do protocolo Ethereum, EOSIO apresenta uma arquitetura que independe de uma imple-

[10]Desde que consenso deve ser necessariamente atingido pela maioria dos participantes da rede, a analogia com o voto também é válida. Em redes PoW, um voto equivale a uma unidade de *hashrate* e em redes PoS, um voto equivale a uma unidade do *token*

[11]Tecnicamente, qualquer pessoa pode competir para minerar um bloco na rede Bitcoin, mas na prática existe uma grande centralização do *hashrate*, que fica sob controle majoritário de poucas grandes empresas.

mentação específica de máquina virtual e o custo da validação das transações é coberto pela inflação anual definida pela constituição, destinada em parte para incentivar os produtores de blocos. Ao possuir os *tokens*, o usuário possui o direito de realizar transações livremente, proporcionalmente ao seu *stake* e, desta forma, introduz-se um modelo eficiente sem a necessidade de taxas de transação como em *chains* PoW, que pode inclusive gerar problemas com o fisco em muitos países.

Por utilizar nativamente contratos compilados em WebAssembly, o desenvolvimento de *dapps* poderá ser feito em múltiplas linguagens. Abaixo um exemplo de como seria uma função de transferência implementada em C++ no contexto de um básico tipo de contrato: uma moeda (*token*) [12].

```
1  void token::transfer( name    from,
2                        name    to,
3                        asset   quantity,
4                        string  memo )
5  {
6      eosio_assert( from != to, "cannot transfer to self" );
7      require_auth( from );
8      eosio_assert( is_account(to), "to account does not exist" );
9      auto sym = quantity.symbol.code();
10     stats statstable( _self, sym.raw() );
11     const auto& st = statstable.get( sym.raw() );
12
13     require_recipient( from );
14     require_recipient( to );
15
16     eosio_assert( quantity.is_valid(), "invalid quantity" );
17     eosio_assert( quantity.amount > 0, "must transfer positive
           quantity" );
18     eosio_assert( quantity.symbol == st.supply.symbol, "symbol
           precision mismatch" );
19     eosio_assert( memo.size() <= 256, "memo has more than 256
           bytes" );
20
21     auto payer = has_auth( to ) ? to : from;
22
23     sub_balance( from, quantity );
24     add_balance( to, quantity, payer );
25 }
```

Ao combinar elevado *throughput* com conveniências para seu público-alvo, os desenvolvedores, EOSIO permite a criação de aplicações atualmente impraticáveis no

[12] Disponível em: ⟨https://github.com/EOSIO/eosio.contracts/blob/master/eosio.token/src/eosio.token.cpp#L87⟩ (acesso em 17-11-2018).

contexto de qualquer outra *blockchain*. Fora que a distribuição da emissão da moeda é mais uniforme, como verificado na figura 2.4.

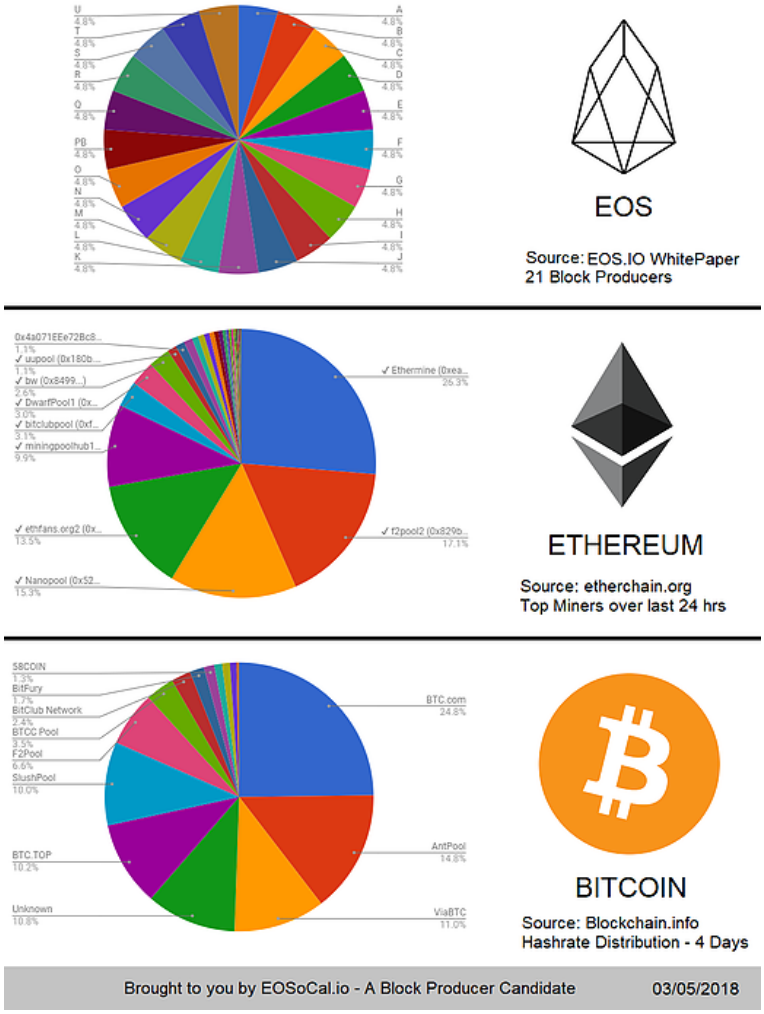


Figura 2.4: Diferença na distribuição de validadores entre DPoS e PoW[13]

### 2.3.1 Governança

Uma característica do projeto é a tentativa em incluir aspectos governamentais numa *blockchain*. Ao contrário de protocolos onde se firma na ideia de que o código-fonte é lei, EOSIO é a primeira *blockchain* a fazer uso de Contratos Ricardianos (GRIGG, 2004). Nestes, os desenvolvedores conseguem delinear a intenção de seus programas de forma que, em caso de *bug* ou disputa entre terceiros, o mesmo sirva como um regulamento para consenso (LARIMER, 2018b).

[13]Disponível em: ⟨https://eossocal.io/⟩ (acesso em 17-11-2018).

Fato é que, mesmo informalmente, processos mais ou menos definidos de governança existem em todas as *blockchains*. Permissões de escrita em repositórios, qual versão rodar, quais ideias implementar e o que fazer em caso de *bugs* são alguns exemplos de situações em que deve haver consenso dos participantes da rede, mesmo que sem um processo de tomada de decisão bem definido (LARIMER, 2018a). Em momentos de crise como na ocasião em que o Ethereum Classic (ETC) foi criado através de um *hardfork* do Ethereum, uma decisão drástica por conta de um *bug*[14] teve de ser tomada e os participantes da rede simplesmente não possuíam os meios para fazê-la consensualmente de forma descentralizada. O resultado surgiu através do voto da Ethereum Foundation. O grupo discordante se viu forçado a separar-se na criação de uma rede independente, uma solução longe do ideal. Tais *forks* contingentes, em sua maioria originados na divergência de opiniões, também aconteceram por diversas vezes na rede Bitcoin.

A tomada de decisão em *blockchains* EOSIO é feita por via de votação por aprovação (FISHBURN, 1978) onde um *token* equivale a um voto. Críticos desse sistema afirmam que a compra de votos inevitavelmente faria o protocolo se degradar para uma plutocracia (BUTERIN, 2018). Mas, tomando-se a figura de uma *blockchain* ser análoga à uma estação de rádio, em que pessoas escutam e transmitem, temos que:

1. Em sistemas PoW como Bitcoin e Ethereum, para definir quem possui a permissão de transmitir parte-se do critério de quem consiga transmitir com maior potência.

2. Em sistemas PoS, exclui-se a necessidade de uma estação grande o suficiente para anular o sinal dos demais. Porém, o requisito técnico de operação ainda existe.

3. Em sistemas DPoS, a comunidade escolhe quais estações terão permissão de transmitir.

Trata-se de uma abordagem que aproxima a tomada de decisão a um processo democrático. É possível que atores maliciosos consigam obter o controle de 51% dos votos, mas há de se levar em consideração o custo da compra destes *tokens* além de que, em última instância, um *hardfork* continua sendo uma opção (LARIMER, 2018c).

[14] No contrato TheDAO.

Em *blockchains* EOSIO, necessita-se de ao menos (2/3)+1 dos BPs para se obter consenso. Na principal rede fazendo uso do protocolo, EOS, conta-se com um grupo de 21 BPs que é mantido em xeque através do poder do voto da comunidade. Esta também pode exigir que BPs atualizem seus nós com modificações do sistema via referendo[15] (até o momento, diversas atualizações menores já foram implementadas, bastando a autorização de 15 BPs). Além disso, a constituição a que todos os usuários da rede ficam submetidos prevê pelo menos um fórum de arbitragem (ECAF), onde disputas são submetidas para orientar a decisão dos BPs. Casos recentes onde fundos ou contas roubados conseguiram retornar a seus donos originais sugerem a aplicabilidade de governança mais próxima da realidade[16]. Pessoas comuns não irão adotar esta nova tecnologia em suas vidas se perder tudo, de forma irreversível, esteja a distância de um clique - como em outras *blockchains*.

## 2.4 Considerações Finais

O protocolo Bitcoin possui grande capacidade de resiliência a ataques diversos e censura, o que justifica sua crescente adoção. Por representar uma alternativa à oligopólios seculares de reserva fracionária, a tecnologia conhecida como *blockchain* inventada por Nakamoto (2008) é o embrião necessário para a criação e manutenção de mercados mais livres, com um viés comunitário onde os próprios participantes são os responsáveis pela geração de valor.

Ao se explorar inovações além do Bitcoin como o protocolo Ethereum, fica nítido como novas formas de organizações socioeconômicas poderão surgir a partir do conceito fundamental de descentralização e *tokenização*. Tal demanda, por si só, também indica o grave problema de escalabilidade atualmente em discussão.

Por conta disso, o projeto EOSIO tem como mote a ideia de "descentralizar tudo", promovendo recursos e alta performance para suprir a demanda de desenvolvedores de *dapps*. Comunidades de serviços descentralizados que competirão com suas contrapartes centralizadas como Uber, Airbnb, Spotify, Github e muitos outros poderão tomar forma ao utilizarem o *software* (GRIGG, 2017).

[15] E.g. ⟨https://eosvotes.io/⟩ (acesso em 17-11-2018).
[16] Qualquer usuário pode abrir uma disputa em ⟨https://eoscorearbitration.io/⟩ (acesso em 17-11-2018).

# 3 Revisão da Literatura

Este capítulo traz um levantamento acerca dos problemas enfrentados por desenvolvedores de *dapps* e alguns exemplos de como foram solucionados. Primeiramente, conceitos básicos sobre pseudoaleatoriedade são colocados para, então, analisar-se métodos para se obter entropia numa *blockchain*. Por fim, diversos exemplos sao abordados.

## 3.1 Pseudoaleatoriedade

A demanda por números aleatórios engloba uma ampla gama de aplicações, não sendo exclusiva aos jogos. Além desta, diversas outras áreas empregam seu uso como por exemplo na estatística, criptografia e simulações computacionais. Apesar de a aplicação da aleatoriedade em si tenha se iniciado milênios atrás em métodos divinatórios (e.g. Tarô, I-Ching, astragalomancia, etc), os primeiros avanços de estudo formal surgiram com Cardano, Galileu e Pascal.

Entretanto, torna-se necessário realizar uma distinção importante. Verdadeira aleatoriedade é impossível de ser alcançada de forma determinística (NEUMANN, 1951) e, portanto, existem duas formas básicas de se gerar números aleatórios: extraindo entropia natural do ambiente por via de captação de ruídos externos como entrada (e.g. *random.org* que utiliza-se de sensores atmosféricos); ou através de algoritmos PRNG.

Nestes, um processo aritmético derivado do que ficou conhecido como método Monte Carlo pode ser empregado para a resolução do problema de geração de números aleatórios uniformemente distribuídos. Trata-se de uma técnica de amostragem sofisticada muito útil em diversos problemas e ciências. O termo foi inicialmente cunhado por Ulam e Neumann, à época pesquisando bombas atômicas em Los Alamos, referenciando o antigo cassino localizado em Mônaco. É também citado como simulação estocástica por diversos autores (ANDERSON, 1999).

A premissa é de que um conjunto de números pseudoaleatórios pode ser computado através de um conjunto inicial preestabelecido (semente) de números uniformemente

distribuídos entre 0 e 1 (JOHNSON, 1956). Já eram realizados esforços de longa data nesse sentido, como por exemplo nas tabelas de Tippett (1927), Kendall e Smith (1939), etc.

Uma consequência direta do determinismo do processo é a impossibilidade de evitar ciclos na *stream* de números. Pela amostra (semente) possuir tamanho finito, a repetição dos números é garantida e a qualidade da sequência é diretamente proporcional ao tamanho do ciclo. Isto foi minimizado por Lehmer (1949) ao propor o método de congruências, onde não existem pequenos ciclos imprevisíveis como em métodos antecessores (JOHNSON, 1956). Os geradores congruentes lineares (LCG) são os mais conhecidos e seguem sendo os mais utilizados PRNGs (com diversas adaptações exploradas), pois são fáceis de compreender e implementar, além de possuir execução linear. Abaixo um exemplo de implementação em Python[17]:

```
1 def lcg(modulus, a, c, seed):
2   while True:
3     seed = (a * seed + c) % modulus
4     yield seed
```

O método é delineado pela variável *seed*. Formalmente, define-se um conjunto $X$ de números pseudoaleatórios da seguinte forma:

$$X_{n+1} = (aX_n + c) \bmod m \tag{3.1}$$

Na equação acima, define-se um gerador Lehmer quando $c = 0$ e um gerador congruente misto quando $c \neq 0$. Este é uma adaptação, que surgiu na década de 60, daquele. A distribuição inicial (semente) é denotada por $X_0$. Uma boa maneira de visualizar o mecanismo é pensar na reta $y = ax + c$ sendo desenhada continuamente mas sendo modulada para trazer o traçado de volta para o quadrado $[0, m]$x$[0, m]$, como detalhado na figura 3.1[18].

**Teorema 3.1.1.** *A sequência definida por X possui período completo (isto é, o ciclo é previsivelmente formado após m iterações, ponto de reinício da sequência) se e somente*

[17]Disponível em: ⟨https://en.wikipedia.org/wiki/Linear_congruential_generator⟩ (acesso em 17-11-2018).

[18]Disponível em: ⟨https://people.sc.fsu.edu/~jburkardt/classes/isc_2009/monte_carlo_sampling.pdf⟩ (acesso em 17-11-2018).

*se:*

1. *o modulo $m$ e o incremento $c$ forem relativamente primos;*
2. *$a - 1$ for divisível por todos os fatores primos de $m$;*
3. *$a - 1$ for divisível por 4 se $m$ for divisível por 4.*

O teorema de Hull e Dobell (1962) exposto acima garante a existência de um subconjunto de permutações entre $a$, $c$ e $m$ que implicam em períodos maximizados. Desta forma, a qualidade da sequência será estatísticamente satisfatória e capaz de ser aprovada em testes de aleatoriedade.

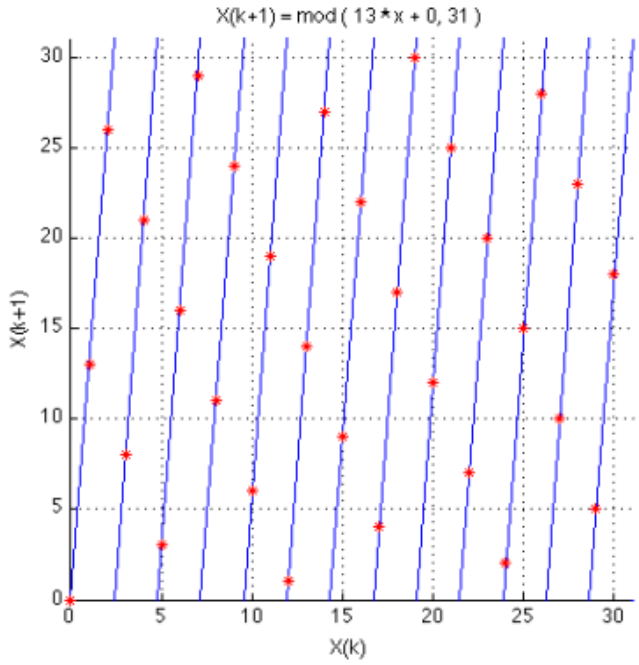


Figura 3.1: Instância de gerador congruente

### 3.1.1 Entropia em Blockchains

Existem quatro métodos básicos atualmente sendo utilizados para a geração de números aleatórios no contexto de *blockchains* (HESS, 2016):

1. O primeiro e mais simples, por não despender muito processamento, consome a entropia inerente aos próprios dados da rede. Quaisquer informação contida no livro razão pode ser arbitrariamente utilizada para esta finalidade.

2. Uma outra maneira surge ao utilizar um oráculo, um tipo de contrato inteligente, responsável por interagir com uma API externa (e.g. *random.org*) capaz de fornecer grande entropia.

3. Também há a ideia de combinar uma semente do cliente com uma do próprio contrato sendo executado. Desta forma, uma *string* aleatória enviada pelo cliente numa transação é utilizada em conjunto de uma outra, enviada por algum outro cliente ou até mesmo extraída da *blockchain* por via do primeiro método pelo próprio contrato.

4. Por fim, Neowiz (2018) propõe uma outra alternativa em que se utiliza uma assinatura criptográfica como entropia. Nele, o ator que requer o número aleatório (*revealer*) interage com o ator que o fornece (*committer*) através de um contrato inteligente, como explicado na figura 3.2.

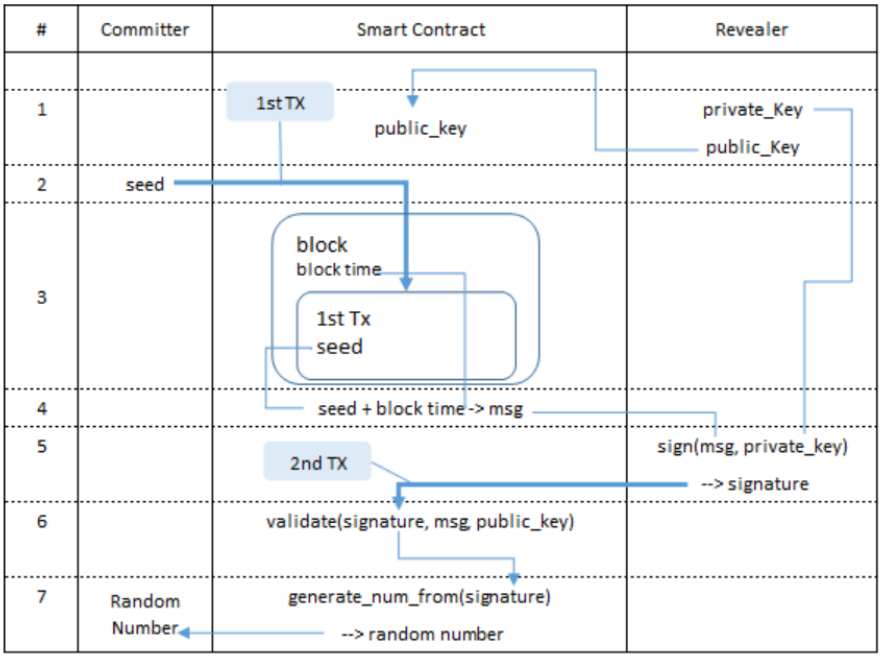


Figura 3.2: Modelo de geração de números pseudoaleatórios via assinaturas criptográficas (NEOWIZ, 2018)

A escolha de um método para uma aplicação específica deve levar em conta algumas implicações. Quanto ao primeiro, é importante considerar que toda e qualquer informação contida numa *blockchain* é de domínio público e além disso passível de *front-running*[19] pelos mineiros, não sendo possível sua adoção em grande parte dos casos de uso

[19]Uma vez que o mineiro tem total controle sobre quais transações serão incluídas num bloco, ele

com demanda por aleatoriedade. Além disso, em *blockchains* que se utilizam do software EOSIO, por exemplo, não é possível extrair dados como *hashes* de blocos em contratos inteligentes.

O segundo garante, a princípio, uma excelente fonte de entropia mas o custo disso é a dependência de um ator externo, aumentanto significativamente a centralização do sistema. Afinal, situações não raras de instabilidade num servidor central como o disponível em *random.org* e afins representam um grande risco para a previsibilidade e determinismo do sistema e, fora isso, é necessário depositar confiança de que tal serviço seja e continue sendo íntegro e honesto.

Atinge-se no terceiro método uma solução mais confiável e próxima do ideal, porém a complexidade do sistema aumenta consideravelmente. Corre-se o risco de reduzir a qualidade da experiência de usuário por conta disso e, de fato, a última técnica é a que mais vem sendo utilizada em *dapps* atuais no ecossistema EOSIO.

## 3.2 Aplicações Relacionadas

Apesar do foco do presente trabalho seja a implementação de um sistema descentralizado que funcione em *blockchains* EOSIO, alguns dos exemplos citados nesta seção foram retirados de outras plataformas, por serem facilmente encontrados na literatura.

### 3.2.1 Definição de Dado Descentralizado

O jogo mais simples que pode ser imaginado de forma descentralizada numa *blockchain* com PoW como mecanismo de consenso simula o funcionamento de um dado virtual: ao apostar num determinado valor, o jogador ganha a partida caso o número de *threshold* aleatoriamente escolhido seja menor que aquele. A probabilidade de ganhar uma partida é representado por $p$, a quantia apostada por $w$ e $h$ representa a comissão do cassino virtual. Piasecki (2016) define o valor esperado de uma partida como:

$$EV = (h - 1)w \tag{3.2}$$

poderia escolher ignorar uma transação vencedora, por exemplo.

Nota-se que o jogador garantidamente irá perder uma pequena parte do montante a cada partida, visto que este é o mecanismo básico que garante o lucro de um cassino. A quantia a ser recebida por um possível ganhador é expressa como PV e o mesmo deve pagar PD em caso de derrota, trivialmente definido como -w. Trata-se do valor em média que o jogador pode esperar receber numa partida individual vitoriosa:

$$EV = (1 - p)PD + pPV$$

Ao utilizar-se da equação 3.2, vem:

$$(h - 1)w = (1 - p)(-w) + pPV$$

$$PV = \frac{wh}{p} - w \tag{3.3}$$

Por fim, temos que a premissa fundamental é de que uma aposta vencedora será recompensada positivamente. Portanto:

$$PV > 0$$

Disto surge um novo limite ao domínio, quando utilizada a equação 3.3:

$$\frac{wh}{p} - w > 0$$

$$h > p \tag{3.4}$$

A forma mais simples e direta de garantir imprevisibilidade para $p$ é utilizar dados da própria *blockchain*, como por exemplo o *hash* de blocos e transações. Porém, por serem informações a princípio manipuláveis por nós mineiros e potenciais atacantes, diversos cenários foram analisados extensivamente. Conclui-se que um contrato inteligente numa rede PoW é o suficiente para garantir a segurança de um jogo como o descrito, para apostas onde $w \leq 1/h$ (PIASECKI, 2016).

### 3.2.2 Exemplos de Dados

A primeira *dapp* a fazer uso da *blockchain* Bitcoin para a implementação de um jogo de dado foi anunciado em Abril de 2012, e continua em operação nos dias de hoje. Satoshi-DICE utiliza o *hash* da transação em que o usuário submete sua aposta em conjunto com uma semente que o mesmo desconhece para fabricar outro *hash* SHA-512. Os primeiros quatro *bytes* configuram o número sorteado (VOORHEES, 2012).

Etheroll é um contrato que simula um dado virtual na plataforma Ethereum. O jogador envia uma transação para a rede com o objetivo de executar a função *playerRollDice(uint rollUnder)* do contrato, que fica a cargo de então executar uma subconsulta ao serviço especializado da *random.org* através do oráculo implementado por *oraclize.it*. De maneira geral, o oráculo fica apenas responsável pela consumação da API, como consta no *script* da função[20] (ETHEROLL, 2016):

```
1 /*
2 * assign partially encrypted query to oraclize
3 * only the apiKey is encrypted
4 * integer query is in plain text
5 */
6 bytes32 rngId = oraclize_query("nested", "[URL] ['json(https://
    api.random.org/json-rpc/1/invoke).result.random[\"serialNumber
    \",\"data\"]', '\\n{\"jsonrpc\":\"2.0\",\"method\":\"
    generateSignedIntegers\",\"params\":{\"apiKey\":${[decrypt]
    BLTr+ZtMOLP2SQVXx8GRscYuXv+3wY5zdFgrQZNMMY3oO/6
    C7OoQkgu3KgfBuiJWW1S3U/+ya10XFGHv2P7MB7VYwFIZd3VOMI/
    Os8o1uJCdGGZgpR0Dkm5QoNH7MbDM0wa2RewBqlVLFGoZX1PJC+igBPNoHC4
    =},\"n\":1,\"min\":1,\"max\":100,\"replacement\":true,\"base
    \":10${[identity] \"}\"},\"id\":1${[identity] \"}\"}']",
    gasForOraclize);
```

No contexto EOSIO, a primeira *dapp* de dado foi implementada pela EOSBet. Apesar de terem lançado alguns protótipos na Ethereum em 2017, consolidaram a decisão de concentrar seus esforços em EOSIO. Os motivos elencados pelos desenvolvedores condiz com o que foi falado neste trabalho. Maior *throughput*, inexistência de taxas de transação (melhorando, portanto, a experiência dos usuários) e maior facilidade de desenvolvimento são os principais.

[20] O código fonte do contrato está disponível em: ⟨https://etherscan.io/address/0xece701c76bd00d1c3f96410a0c69ea8dfcf5f34e#code⟩ (acesso em 17-11-2018).

### 3.2.3 Exemplos de Loterias

Uma plataforma que oferece uma loteria descentralizada, conhecida como Trueflip, recompensa apostas certeiras de números num determinado intervalo. Leva o prêmio quem acertar 5 números dentre 49 e 1 número dentre 26. Para a geração aleatória dos números sorteados, adota-se um processo que se baseia no primeiro método abordado na última seção. Por definição, o contrato decodifica o *hash* do primeiro bloco válido que tenha sido minerado após um *timestamp* específico, no protocolo Bitcoin. Existe uma janela de tempo de forma que o cadastro de apostas se encerra antes deste momento (TRUEFLIP, 2017). Por exemplo, ao considerar o *hash* do bloco #463346:

$$000000000000000001bf3b0a7fdf7e4889734f4f0c0cb00a1bb082cf3d4ad49$$

Para a extração de 6 números, divide-se o final do *hash* em segmentos de 4 caracteres. Cada segmento $x_i$ é modulado e decodificado de hexadecimal para decimal:

$$x_0 = (f0c0)_{16} \bmod 49 = (61632)_{10} \bmod 49 = 39$$

$$x_1 = (cb00)_{16} \bmod 49 = (51968)_{10} \bmod 49 = 28$$

$$x_2 = (a1bb)_{16} \bmod 49 = (41403)_{10} \bmod 49 = 47$$

$$x_3 = (082c)_{16} \bmod 49 = (2092)_{10} \bmod 49 = 34$$

$$x_4 = (f3d4)_{16} \bmod 49 = (62420)_{10} \bmod 49 = 43$$

$$x_5 = (ad49)_{16} \bmod 49 = (44361)_{10} \bmod 26 = 5$$

Diferentemente da Trueflip, a loteria Kibo recolhe sementes geradas no *frontend* da aplicação pelos jogadores no momento de compra dos bilhetes que garantem a participação do jogo e os mistura com *hashes* de blocos recentes. Esta aplicação é peculiar pois sugere mecanismos de *marketing* e governança não oferecidos nativamente pelo protocolo Ethereum. Sendo na realidade um conjunto de contratos, Kibo define que a gestão destes deva ser realizada à partir de decisões tomadas por via de votação de quem possui *stake* do Control Token (KIBO, 2016).

### 3.2.4 Exemplos Diversos

Um exemplo interessante é o contrato RANDAO, que objetiva fornecer números aleatórios. Nele, um esforço colaborativo é modelado num processo que envolve três etapas[21]:

1. Coleção de *hashes sha3*: qualquer usuário interessado em participar na geração de um número aleatório envia ao contrato uma quantia $m$ de Ether como garantia e o resultado de $sha3(s)$, sendo $s$ um número secreto escolhido arbitrariamente pelo mesmo como semente.

2. Validação das sementes: todos os participantes inclusos na primeira etapa devem agora enviar ao contrato suas respectivas sementes dentro de um período de tempo previamente estipulado. O contrato faz a verificação criptográfica e monta a coleção de sementes validadas.

3. Cálculo do número: após a montagem da coleção, o contrato executa uma função $f(s_0, s_1, ..., s_n)$ que resulta num número aleatório que é, então, enviado de volta para os participantes junto do valor $m + L$ em Ether. O lucro ($L$) do contrato ocorre das taxas paga por outros contratos ao solicitar aleatoriedade, que é então dividido proporcionalmente à $m$ aos participantes do processo.

Outro exemplo é a *blockchain* Peerplays, que foi construída utilizando o *software* Graphene (precursor do EOSIO, também criado por Daniel Larimer) e, portanto, utiliza como mecanismo de consenso o DPoS. Jogadores interagem com a plataforma através de contratos inteligentes e fazem uso de um PRNG embutido. Para todo prêmio recebido, a rede cobra uma pequena porcentagem que retorna como lucro para três direções. A maior parte é destinada para todos os endereços que possuem o *token* nativo da plataforma como uma espécie de dividendo, outra parte para um fundo que arca com eventuais custos de desenvolvimento e resolução de *bugs* e por último um fundo a ser sorteado em frequentes Mega-Jackpots (BAHA'I; MALONEY, 2017).

Fora do contexto de cassinos, aplicações como FirstBlood e Unikrn (2017) idealizam a descentralização de apostas em partidas e torneios de *eSports* como por exemplo

[21] Explicação disponível em: ⟨https://github.com/randao/randao/blob/master/README.md⟩ (acesso em 17-11-2018).

League of Legends, DOTA 2 e Counter-Strike. No caso da primeira, é proposto um sistema de verificação de resultados pois a não ser que o jogo esteja rodando na EVM, será necessário um mecanismo de consenso para que os mesmos sejam inseridos na *blockchain* Ethereum. Para tanto, é preciso que no mínimo duas testemunhas, que rodam *software* especializado em conexão com APIs centralizadas (tal como um oráculo) que fazem o *broadcast* oficial das estatísticas de uma partida, cheguem a um consenso do resultado. Caso não seja possível, um júri é formado para determinar o resultado original através de um sistema de votação. O processo de seleção de testemunhas e componentes do júri é aleatório e requer *stake* do *token* nativo e além disso, para evitar ataques *sybil* onde contas são criadas como *spam* objetivando vantagens para o operador, a aplicação contará com o suporte de um banco de dados centralizado onde são guardadas informações de identidade dos agentes citados (CUESTA et al., 2016).

Muitas *dapps* vêm surgindo no contexto EOSIO. Alguns exemplos incluem a HireVibes[22], que é um aplicativo que permite que empregadores encontrem empregados; Nougit[23], que é um sistema de gerenciamento de repositórios git que faz uso de IPFS e valida os *hashes* na *blockchain*; e por fim a Everipedia[24], que é uma concorrente da Wikipedia construída em EOSIO, que conta com um cofundador desta no time, Larry Sanger.

## 3.3 Considerações Finais

Apesar de possibilitar a implementação de *dapps* que rodam numa máquina virtual distribuída como a EVM, a eficiência da mesma é crucial na maior parte de casos de uso reais. Caso modelado despreocupadamente com lenta computação dos estados, um problema não será solucionado a ponto de ficar utilizável, como verificado por Grau (2016) num jogo de xadrez que precisou passar por reformulações e recorrer à computação *off-chain* a cargo dos clientes para viabilizar seu uso na EVM; ou no caso da *dapp* VGO que abandonou a Ethereum apenas 72 horas após seu lançamento, tendo escolhido uma *blockchain* DPoS para abrigar seu negócio (WAX, 2018).

[22]Disponível em: ⟨https://hirevibes.io/⟩ (acesso em 17-11-2018).
[23]Primeiro lugar na EOS Global Hackathon em São Francisco, 12/11/2018.
[24]Disponível em: ⟨https://everipedia.org/⟩ (acesso em 17-11-2018).

# 4 Solução Proposta

Este capítulo descreve a loteria proposta pelo presente trabalho. Faz-se necessária uma contextualização acerca do desenvolvimento e operação de contratos com o protocolo EOSIO. Na subseção seguinte, o funcionamento da loteria é visto em detalhes e, por fim, algumas considerações pertinentes são apresentadas.

## 4.1 Contratos em EOSIO

Para se desenvolver contratos para *blockchains* EOSIO, uma SDK - ou CDT - foi criada[25]. O CDT auxilia os desenvolvedores ao fornecer uma série de ferramentas essenciais, como: biblioteca eosiolib, compilador[26] e gerador de ABI. O exemplo mais simples de contrato como visto abaixo é compilado em um binário (WebAssembly), para que possa ser serializado e implantado numa *blockchain*.

```
1  #include <eosiolib/eosio.hpp>
2
3  using namespace eosio;
4  using namespace std;
5
6  class [[eosio::contract("hello")]] hello : public contract
7  {
8      public:
9          using contract::contract;
10
11         [[eosio::action]] void printact( string s )
12         {
13             print( s );
14         }
15 };
16
17 EOSIO_DISPATCH( hello, (printact) )
```

Os atributos da biblioteca eosiolib ([[eosio::contract()]] e [[eosio::action]]) indicam ao gerador de ABI o que deve ser incluído. Descrita em formato JSON, esta interface também é implantada na *blockchain* e serve como referência para que os nós da rede

[25]Disponível em: ⟨https://github.com/eosio/eosio.cdt/⟩ (acesso em 17-11-2018).
[26]Clang modificado com suporte experimental a WebAssembly.

possam interagir com o contrato, além de delinear o contrato Ricardiano para cada ação. A macro EOSIO_DISPATCH implementa a função *apply*, que é chamada por *default* pelos BPs na execução de qualquer ação de um contrato. Desta forma, abstrai-se o código necessário do protoco, que fica encapsulado, e um *switch* interno encaminha o fluxo para o método da ação, neste caso o *printact*. Além da ação, a conta em que foi implantado o código e a conta que autorizou a chamada fazem parte dos argumentos de *apply*.

Para compilar e gerar a ABI do contrato utiliza-se os dois comandos:

```
% eosio-cpp -contract=hello -o=hello.wasm hello_world.cpp
% eosio-abigen -contract=hello -output=hello.abi hello_world.cpp
```

Isto feito, para se implantar o código hello.wasm e sua ABI hello.abi, bem como para enviar outros comandos para os nós - ou *nodeos* - da rede[27], é necessário a utilização de uma interface de usuário. Essa interface consome uma API HTTP, arquitetada em RPC e exposta pelo *nodeos*, sendo que a mais simples é a *cleos* (CLI). Ambos pertencem ao *software* EOSIO[28].

```
[00:15:44 eos@eos:~/git/hello_world]
% cleos set code accountnum11 hello.wasm
Reading WASM from hello.wasm...
Setting Code...
executed transaction: 41bbbb585b83fe95b9249cad2fd35b5a31beeb1d69d024e3b55cc74cd958ff08  280
0 bytes  678 us
#         eosio <= eosio::setcode               {"account":"accountnum11","vmtype":0,"vmver
sion":0,"code":"0061736d01000000015c1060027f7f00600000600...
warn  2018-11-15T00:15:49.957 thread-0  main.cpp:482                  print_result
warning: transaction executed locally, but may not be confirmed by the network yet

[00:15:49 eos@eos:~/git/hello_world]
% cleos set abi accountnum11 hello.abi
Setting ABI...
executed transaction: 7dd0fccb8585f3fae336cb1c0bd950683007ff2a130e5121847105badd5718cd  144
 bytes  601 us
#         eosio <= eosio::setabi                {"account":"accountnum11","abi":"0e656f7369
6f3a3a6162692f312e3000010870726 96e7461637400010173067 3747...
warn  2018-11-15T00:15:54.705 thread-0  main.cpp:482                  print_result
warning: transaction executed locally, but may not be confirmed by the network yet

[00:15:54 eos@eos:~/git/hello_world]
% cleos push action accountnum11 printact '["hello world"]' -p accountnum11
executed transaction: 0b59ca1eb0292fbffe43eaaff2ebb010dbaf7162b11482640d0bc7c4e7ac54e2  104
 bytes  556 us
#  accountnum11 <= accountnum11::printact       {"s":"hello world"}
>> hello world
warn  2018-11-15T00:15:58.802 thread-0  main.cpp:482                  print_result
warning: transaction executed locally, but may not be confirmed by the network yet
```

Figura 4.1: Implantação de um contrato e sua ABI

[27]Mais detalhes disponíveis em: ⟨https://developers.eos.io/⟩ (acesso em 17-11-2018).
[28]Disponível em: ⟨https://github.com/eosio/eos/⟩ (acesso em 17-11-2018).

As chamadas eosio::setcode[29] e eosio::setabi são ações do contrato do sistema (*eosio.system*). Além deste, outros integram o sistema como por exemplo *eosio.token* (que implementa a moeda da rede) e *eosio.msig* (*multisig*, para que BPs autorizem ações em consenso). Tanto as chamadas a ações, de qualquer contrato, quanto o *output* destas são codificados em formato JSON, definindo assim uma mensagem. Mensagens são trocadas pelos nós, que mantém o estado final da rede em memória e a depender de *plugins* podem replicar o conteúdo em bancos como MariaDB ou MongoDB. Por possuir uma arquitetura modular, com *plugins* para customizar o funcionamento do *nodeos* e foco em contratos, os desenvolvedores optaram por desacoplar as funcionalidades básicas da camada de consenso.

Ações como criar novas contas, votar em BPs, delegar *stake* e comprar ou vender memória RAM são algumas do *eosio.system*. Cada ação, ao ter sua mensagem transmitida, consome banda (NET). Quanto maior o tamanho da mensagem, mais dados terão que ser transferidos. Além da demanda de transmissão, existe a demanda por processamento em cada ação (CPU), a depender do método que ela fará os BPs executarem. Quanto maiores os *stakes* (CPU ou NET) de uma conta autorizando a execução de ações, mais recursos ela pode consumir. Qualquer conta pode delegar CPU e NET para outras contas[30], sendo que o total *staked* por uma conta define seus votos - caso ela tenha votado.

Um outro atributo da eosiolib é o [[eosio::table]], usado na definição de tabelas que por sua vez são definidas em *structs*. Elas servem para que um contrato mantenha em memória o estado de seus dados[31]. Os contratos fazem leituras e escritas como num banco de dados, por via de *containers* MultiIndex da biblioteca Boost. Contratos também podem ler tabelas de outros contratos, sendo que esta comunicação intercontratual deve ser considerada assíncrona. Este modelo pode resultar em *spam*, mas o algoritmo de limitação dos recursos resolve. Segue um exemplo de tabela com apenas uma coluna e uma operação de inserção para ilustrar:

[29] Novas versões do binário podem ser reimplantadas, efetivamente atualizando o código do contrato que é disponível para os nós da rede. Isto não é possível na rede Ethereum.

[30] É possível, inclusive, emprestar recursos a outras contas usando *dapps* como Chintai - ⟨https://chintai.io/⟩ (acesso em 17-11-2018). Uma solução nativa num novo contrato de sistema está previsto para uma próxima versao do EOSIO.

[31] Por exemplo, o balanço de moedas que uma conta possui é mantido numa tabela - a diferença fundamental para com o modelo UTXO.

```
1  struct [[eosio::table]] player
2  {
3      name player;
4      auto primary_key() const { return player.value; }
5  };
6  multi_index< name( "player" ), player > players;
7
8  [[eosio::action]] void insert( name row )
9  {
10     players.emplace( _self, [&]( auto& p ) { p.player = row; } );
11 }
```

Chamar a ação *insert* faz com que o contrato insira uma nova linha em sua tabela *player* com o valor *row*. Além de *emplace*, diversos outros métodos são fornecidos pela API MultiIndex como por exemplo o *find* que serve para buscar um registro de acordo com a chave primária da tabela. Outros índices também podem ser criados numa mesma tabela, além de ser possível definir uma tabela com mais colunas - inclusive de outros tipos.

Enquanto não for removida, uma linha consumirá a quantidade necessária de memória RAM de acordo com sua estrutura. O código do contrato também é mantido em memória. Por ser um recurso finito, a memória deve ser comprada e vendida - ao invés de seguir a lógica de *stake* como nos casos de CPU e NET. Para tanto, a conta interessada deve interagir com o contrato do sistema (*eosio.system*) via ações *buyram* e *sellram*. Não existe um mercado propriamente dito por se tratar de um relé Bancor, responsável por precificar a quantidade disponível de RAM de acordo com a oferta e a demanda.

## 4.2 Modelo da Aplicação

A *dapp* proposta conta com dois contratos: o de um *token* e o do jogo propriamente dito. A fonte do *token* é um *fork* do *eosio.token*, mas customizado. Na transferência, como visto na seção 2.3, foi acrescentada uma condição que verifica se o destinatário dos *tokens* é a conta com o código do jogo. Em caso positivo, uma ação no contrato do jogo é desencadeada, na mesma transação, para inserir um novo jogador. Desta forma, a lógica do jogo diz que quanto mais *tokens* uma conta apostar, maiores suas chances de ganhar o prêmio. As ações do contrato da loteria constam abaixo:

```
1  [[eosio::action]] void setgame( uint64_t max_players,
2                                  uint32_t interval,
3                                  uint8_t  stage );
4
5  [[eosio::action]] void reset( name reset );
6
7  [[eosio::action]] void submithash( name             player,
8                                     asset            quantity,
9                                     capi_checksum256 hash );
10
11 [[eosio::action]] void submitboth( name             player,
12                                    capi_checksum256 hash,
13                                    capi_checksum256 secret );
```

Um contrato pode executar outras ações, inclusive uma de si mesmo - recursividade é suportada - em duas formas: *inline*, que garante a atomicidade da transação e se alguma de suas ações falhar todas falham, ou *referred*, que vai para uma fila de baixa prioridade para BPs executarem quando lhe for pertinente e assim economizar CPU.

O jogo possui duas tabelas: *game* e *player*, especificadas abaixo:

```
1  struct [[eosio::table]] game
2  {
3      time_point_sec deadline;
4      uint64_t       max_players;
5      uint32_t       interval;
6      uint8_t        stage;
7      asset          total_pot;
8  };
9
10 struct [[eosio::table]] player
11 {
12     name             player;
13     capi_checksum256 hash;
14     capi_checksum256 secret;
15     asset            quantity;
16     boolean          active;
17     auto primary_key() const { return player.value; }
18 };
```

A tabela *game* não precisa de nenhum índice pois é usada como *singleton* pelo contrato, para manter o estado atual do jogo e, portanto, necessita de apenas uma linha. O jogo é feito em dois estágios: no primeiro, novos jogadores - ao transferirem *tokens* para a conta do contrato - executam a ação *submithash*; no segundo, novos jogadores não são aceitos e os que foram registrados anteriormente devem executar a ação *submitboth*.

Por ter sido modificada a ação de transferência do *token* para executar a *submithash* do jogo de maneira *inline*, utiliza-se da função *require_auth* da eosiolib para permitir que somente o contrato do *token* execute esta ação. Desta forma, a única maneira de novos jogadores se registrarem no primeiro estágio é transferindo *tokens*. Outra restrição diz respeito ao conteúdo do campo *memo* desta transferência. Deve ser um *hash* SHA-256 que será enviado como o terceiro parâmetro da *inline*. Trata-se da *seed* do jogador que será revelada no segundo estágio para poder ser usada como entropia na escolha dos vencedores, como visto no terceiro método da seção 3.1.1.

Um novo registro é então criado na tabela *player*, que é removido apenas no término do jogo. Cada linha da tabela irá consumir aproximadamente 0.2 KiB com essa estrutura e, portanto, deve-se limitar a quantidade permitida de jogadores (*game.max_players*) de acordo com a disponibilidade de memória RAM na conta do jogo.

A computação da lógica do jogo fica a cargo da ação *reset*, que pode ser executada por qualquer usuário da rede a qualquer momento. São sorteados três ganhadores e se no segundo estágio não houver pelo menos quatro jogadores ativos (que chamaram com sucesso ambas *submithash* e *submitboth*), o jogo reembolsa. O atributo *game.stage* que define o estágio atual do jogo também é controlado pela ação *reset* e é alterado apenas se *now()* for maior que *game.deadline*, sendo que cada estágio dura *game.interval* segundos. Para montar o número pseudoaleatório, uma lista encadeada numa estrutura auxiliar (*prng*) foi utilizada. Assim, para cada jogador o *hash* anterior do número em construção é misturado com os dados em sua estrutura *player*. No final é definido um *hash* formado em conjunto por todos os jogadores, que é utilizado como o número sorteado. O procedimento é ilustrado abaixo:

```
1 capi_checksum256 _seed;
2 sha256( 0, 0, &_seed );
3 auto seed = prng{ _seed, active_players.front() };
4 for( player p : active_players ) {
5     seed.next = p;
6     sha256( ( char* ) &seed, sizeof( prng ), &seed.last );
7 }
8 capi_checksum256 prn = seed.last;
```

## 4.3 Considerações Finais

Uma interface *web* foi construída para tornar possível uma experiência de usuário facilitada. Fez-se uso de duas bibliotecas JavaScript para a integração: eosjs e scatter-js. A primeira recupera os dados da *blockchain* e a segunda permite que os jogadores autorizem as ações com suas chaves privadas por via do Scatter.

Um grande problema de jogos virtuais de forma geral é a utilização de *scripts*, ou *bots*, que interagem com o sistema de forma automâta. O artifício de CAPTCHAs, apesar de ser aplicável, não seria muito eficaz pois os *bots* podem interagir diretamente com o contrato sem utilizar a interface *web*. O importante é que todas as ações e resultados do jogo são transparentemente verificáveis, o que o torna justo pois, além do mais, a técnica de extração entrópica garante a satisfatoriedade estatística do jogo.

# 5 Conclusão

É cada vez mais nítido o enorme potencial disruptivo que as *blockchains* representam para inúmeras indústrias. Ao remover da equação o elemento intermediário com autoridade centralizada e exclusiva sobre os domínios de uma aplicação, muitas instituições tornam-se simplesmente obsoletas neste novo paradigma. O caso mais simbólico é o do sistema bancário de maneira geral, que se vê diante de um grande dilema que só conseguirá ser superado através de enormes esforços inovativos. Ocasiões como o Plano Collor demonstram que o dinheiro presente numa conta está na realidade em posse do banco, na qual se delega a autoridade dos fundos para que o banco faça o que quiser com ele - inclusive, por conta do modelo de reserva fracionária em que o banco é obrigado por lei a manter em reserva apenas uma fração de seus depósitos, emprestar para outros clientes a taxas muito maiores àquela que lhe é recompensada. Curiosamente, a primeira *blockchain* surgiu à época da crise econômica de 2008, que se iniciou nos Estados Unidos mas escalou para todo o mundo, na qual os maiores bancos do mundo[32] foram culpados.

```
00000000   01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000010   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000020   00 00 00 00 3B A3 ED FD  7A 7B 12 B2 7A C7 2C 3E   ....;£íýz{.²zÇ,>
00000030   67 76 8F 61 7F C8 1B C3  88 8A 51 32 3A 9F B8 AA   gv.a.È.Ãˆ ŠQ2:Ÿ¸ª
00000040   4B 1E 5E 4A 29 AB 5F 49  FF FF 00 1D 1D AC 2B 7C   K.^J)«_Iÿÿ...¬+|
00000050   01 01 00 00 00 01 00 00  00 00 00 00 00 00 00 00   ................
00000060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 FF FF  FF FF 4D 04 FF FF 00 1D   ......ÿÿÿÿM.ÿÿ..
00000080   01 04 45 54 68 65 20 54  69 6D 65 73 20 30 33 2F   ..EThe Times 03/
00000090   4A 61 6E 2F 32 30 30 39  20 43 68 61 6E 63 65 6C   Jan/2009 Chancel
000000A0   6C 6F 72 20 6F 6E 20 62  72 69 6E 6B 20 6F 66 20   lor on brink of
000000B0   73 65 63 6F 6E 64 20 62  61 69 6C 6F 75 74 20 66   second bailout f
000000C0   6F 72 20 62 61 6E 6B 73  FF FF FF FF 01 00 F2 05   or banksÿÿÿÿ..ò.
000000D0   2A 01 00 00 00 43 41 04  67 8A FD B0 FE 55 48 27   *....CA.gŠý°þUH'
000000E0   19 67 F1 A6 71 30 B7 10  5C D6 A8 28 E0 39 09 A6   .gñ¦q0·.\Ö¨(à9.¦
000000F0   79 62 E0 EA 1F 61 DE B6  49 F6 BC 3F 4C EF 38 C4   ybàê.aÞ¶Iö¼?Lï8Ä
00000100   F3 55 04 E5 1E C1 12 DE  5C 38 4D F7 BA 0B 8D 57   óU.å.Á.Þ\8M÷º..W
00000110   8A 4C 70 2B 6B F1 1D 5F  AC 00 00 00 00           ŠLp+kñ._¬....
```

Figura 5.1: Conteúdo do primeiro bloco - gênesis - da rede Bitcoin, com referência à manchete do dia 03 de Janeiro de 2009 do jornal inglês The Times: "Tesouro britânico à beira do segundo resgate para os bancos". Os governos inflacionaram suas moedas via artifícios como Quantitative Easing para cobrir a dívida dos bancos insolventes.

Porém, por se tratar de uma invenção recente, sua existência ainda encontra-se longe do radar da maioria dos usuários conectados em rede. De forma análoga, a própria Internet representou uma verdadeira mudança de paradigmas quando foi finalmente ex-

[32] Após 158 anos em operação, o quarto maior banco de investimentos do mundo, Lehman Brothers, foi à falência em 2008.

pandida para além das comunidades acadêmicas e outros nichos mais específicos, atingindo a população geral a partir da segunda metade da década de 1990. Porém, cabe ressaltar que a mesma precisou ficar, de certa forma, incubada por muitos anos até que alcançasse o estado de maturidade necessário para ser adotada em massa[33].

A aplicabilidade das *blockchains* é diversa indo além de somente suportar o funcionamento de moeda de troca e apenas os anos que seguirão vão poder confirmar sua verdadeira utilidade. Acredita-se que cada vez mais casos de uso serão implementados neste novo paradigma descentralizado por conta do argumento de incentivo econômico. Ao possibilitar que os membros de uma comunidade que tanto criam quanto extraem valor da mesma criem um laço econômico eficiente entre si sem a necessidade de intermediários, tornam-se mais próximos da figura clássica de um *stakeholder* e têm seus interesses alinhados. Assim, novas formas de organizações sociais e econômicas surgirão.

Entretanto, muitos desafios ainda se encontram com solução em aberto para a maioria das *blockchains*. Especificamente no âmbito do protocolo Bitcoin, muitos fatores devem ser levados em consideração na tentativa de realizar uma projeção de seu sucesso futuro. Seu algoritmo de força bruta, também adotado ainda na maioria das outras *blockchains*, é responsável por incentivar o consumo desenfreado de energia elétrica pelos nós mineiros, o que pode acabar por torná-lo ecologicamente insustentável no longo prazo. Seu modelo de cobrar taxas em cada transação, apesar de prática comum em todos os sistemas de pagamento mais utilizados até então, é criticado desde por questões financeiras até pela simples experiência de usuário. Potenciais problemas como a quebra das funções criptográficas utilizadas com o advento da computação quântica não podem ser ignorados numa análise mais ampla. Entretanto, a falta de um plano concreto para escalar a *blockchain* para quantos usuários necessários, a inexistência de governança para ditar as regras de implementação de inovações exigidas pela comunidade e a centralização do *hashrate* em pouquíssimas empresas mineradoras são os problemas mais pertinentes.

[33] E.g. não é mais necessário conhecimento técnico de protocolos como TCP/IP ou VoIP para utilizá-los.

# Bibliografia

ANDERSON, E. C. Monte carlo methods and importance sampling. In: . [s.n.], 1999. Acesso em 17-11-2018. Disponível em: ⟨http://ib.berkeley.edu/labs/slatkin/eriq/classes/guest_lect/mc_lecture_notes.pdf⟩.

ANGEL, J. J.; MCCABE, D. M. The business ethics of short selling and naked short selling. In: . [s.n.], 2008. Acesso em 17-11-2018. Disponível em: ⟨https://link.springer.com/article/10.1007%2Fs10551-008-9943-5⟩.

BACK, A. Hashcash - a denial of service counter-measure. In: . [s.n.], 2002. Acesso em 17-11-2018. Disponível em: ⟨http://www.hashcash.org/papers/hashcash.pdf⟩.

BAHA'I, J.; MALONEY, M. P. Peerplays: A provably fair blockchain-based gaming platform. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨https://steemit.com/peerplays/@peerplays/peerplays-whitepaper⟩.

BUTERIN, V. A next-generation smart contract and decentralized application platform. In: . [s.n.], 2014. Acesso em 17-11-2018. Disponível em: ⟨https://github.com/ethereum/wiki/wiki/White-Paper⟩.

BUTERIN, V. Ethereum: Platform review. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨https://bit.ly/1Uv0Mk0⟩.

BUTERIN, V. Governance, part 2: Plutocracy is still bad. In: . [s.n.], 2018. Acesso em 17-11-2018. Disponível em: ⟨https://vitalik.ca/general/2018/03/28/plutocracy.html⟩.

BUTERIN, V.; GRIFFITH, V. Casper the friendly finality gadget. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨https://arxiv.org/pdf/1710.09437.pdf⟩.

CHAUM, D. Blind signatures for untraceable payments. In: . [s.n.], 1982. Acesso em 17-11-2018. Disponível em: ⟨http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF⟩.

COHEN, B. Incentives build robustness in bittorrent. In: . [S.l.: s.n.], 2003.

CORE, B. Segregated witness benefits. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨https://bitcoincore.org/en/2016/01/26/segwit-benefits/#script-versioning⟩.

CUESTA, M. et al. A decentralized esports platform based on smart contracts. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨https://coss.io/documents/white-papers/first-blood.pdf⟩.

DAI, W. b-money. In: . [s.n.], 1998. Acesso em 17-11-2018. Disponível em: ⟨http://www.weidai.com/bmoney.txt⟩.

EASTLAKE, D.; HANSEN, T. Us secure hash algorithms (sha and sha-based hmac and hkdf). In: . [S.l.]: Internet Engineering Task Force (IETF), 2011.

ETHEROLL. Etheroll dice game whitepaper. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨http://crowdfund.etheroll.com/etheroll-whitepaper.pdf⟩.

FINNEY, H. Rpow - reusable proofs of work. In: . [s.n.], 2004. Acesso em 17-11-2018. Disponível em: ⟨http://nakamotoinstitute.org/rpow/⟩.

FISHBURN, S. J. B. . P. C. Approval voting. In: . [s.n.], 1978. Acesso em 17-11-2018. Disponível em: ⟨https://doi.org/10.2307/1955105⟩.

GRAU, P. Lessons learned from making a chess game for ethereum. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨https://medium.com/@graycoding/lessons-learned-from-making-a-chess-game-for-ethereum-6917c01178b6⟩.

GRIGG, I. The ricardian contract. In: . [s.n.], 2004. Acesso em 17-11-2018. Disponível em: ⟨http://iang.org/papers/ricardian_contract.html⟩.

GRIGG, I. Eos: An introduction. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨http://iang.org/papers/EOS_An_Introduction.pdf⟩.

HESS, T. How can i securely generate a random number in my smart contract? In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨https://ethereum.stackexchange.com/questions/191/how-can-i-securely-generate-a-random-number-in-my-smart-contract⟩.

HUGHES, E. A cypherpunk's manifesto. In: . [s.n.], 1993. Acesso em 17-11-2018. Disponível em: ⟨https://www.activism.net/cypherpunk/manifesto.html⟩.

HULL, T. E.; DOBELL, A. R. Random number generators. In: . [s.n.], 1962. Acesso em 17-11-2018. Disponível em: ⟨https://dspace.library.uvic.ca:8443/bitstream/handle/1828/3142/Random_Number_Generators.pdf⟩.

JAKOBSSON, M.; JUELS, A. Proofs of work and bread pudding protocols. In: . [S.l.: s.n.], 1999. ISBN 978-0-387-35568-9.

JOHNSON, D. L. Generating and testing pseudo random numbers on the ibm type 701. In: . [s.n.], 1956. Acesso em 17-11-2018. Disponível em: ⟨https://www.ams.org/journals/mcom/1956-10-053/S0025-5718-1956-0076467-X/S0025-5718-1956-0076467-X.pdf⟩.

KENDALL, M. G.; SMITH, B. B. Random sampling numbers. In: . [S.l.]: Tracts for Computers 24 Cambridge Uni v. Press London, 1939.

KIBO. Kibo lotto whitepaper. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨https://kiboplatform.net/_alldata/files/kibowhitepaper.pdf⟩.

KING, S.; NADAL, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. In: . [s.n.], 2012. Acesso em 17-11-2018. Disponível em: ⟨https://peercoin.net/assets/paper/peercoin-paper.pdf⟩.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. In: . [s.n.], 1982. Acesso em 17-11-2018. Disponível em: ⟨http://people.cs.uchicago.edu/~shanlu/teaching/33100_wi15/papers/byz.pdf⟩.

LARIMER, D. The hidden costs of bitcoin. In: . [s.n.], 2013. Acesso em 17-11-2018. Disponível em: ⟨https://letstalkbitcoin.com/is-bitcoin-overpaying-for-false-security⟩.

LARIMER, D. *Delegated Proof-of-Stake (DPOS)*. 2014. Acesso em 17-11-2018. Disponível em: ⟨https://steemit.com/bitshares/@testz/bitshares-history-delegated-proof-of-stake-dpos⟩.

LARIMER, D. Blockchain utxo model is a dead end for general purpose applications. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨https://steemit.com/blockchain/@dantheman/blockchain-utxo-model-is-a-dead-end-for-general-purpose-applications⟩.

LARIMER, D. Eos.io technical white paper. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md⟩.

LARIMER, D. Decentralized blockchain governance. In: . [s.n.], 2018. Acesso em 17-11-2018. Disponível em: ⟨https://medium.com/@bytemaster/decentralized-blockchain-governance-743f0273bf5a⟩.

LARIMER, D. The "intent of code"is law. In: . [s.n.], 2018. Acesso em 17-11-2018. Disponível em: ⟨https://medium.com/@bytemaster/the-intent-of-code-is-law-c0e0cd318032⟩.

LARIMER, D. The limits of crypto-economic governance. In: . [s.n.], 2018. Acesso em 17-11-2018. Disponível em: ⟨https://medium.com/@bytemaster/the-limits-of-crypto-economic-governance-9362b8d1d5aa⟩.

LEHMER, D. H. Mathematical methods in large-scale computing units. In: . [S.l.: s.n.], 1949.

LUBIN, J. The issuance model in ethereum. In: . [s.n.], 2014. Acesso em 17-11-2018. Disponível em: ⟨https://blog.ethereum.org/2014/04/10/the-issuance-model-in-ethereum/⟩.

MERKLE, R. Protocols for public key cryptosystems. In: . [s.n.], 1980. Acesso em 17-11-2018. Disponível em: ⟨http://www.merkle.com/papers/Protocols.pdf⟩.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. In: . [s.n.], 2008. Acesso em 17-11-2018. Disponível em: ⟨https://bitcoin.org/bitcoin.pdf⟩.

NEOWIZ. Rng in blockchain. In: . [s.n.], 2018. Acesso em 17-11-2018. Disponível em: ⟨https://www.nblab.io/rng⟩.

NEUMANN, J. von. Various techniques used in connection with random digits. In: . [s.n.], 1951. Acesso em 17-11-2018. Disponível em: ⟨https://dornsifecms.usc.edu/assets/sites/520/docs/VonNeumann-ams12p36-38.pdf⟩.

PIASECKI, P. J. Gaming self-contained provably fair smart contract casinos. In: . [s.n.], 2016. Acesso em 17-11-2018. Disponível em: ⟨http://ledger.pitt.edu/ojs/index.php/ledger/article/download/29/53⟩.

POELSTRA, A. On stake and consensus. In: . [s.n.], 2015. Acesso em 17-11-2018. Disponível em: ⟨https://download.wpsoftware.net/bitcoin/pos.pdf⟩.

POON, J.; BUTERIN, V. Plasma: Scalable autonomous smart contracts. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨https://www.plasma.io/plasma.pdf⟩.

SCHUH, F.; LARIMER, D. Bitshares 2.0, financial smart contract platform. In: . [s.n.], 2015. Acesso em 17-11-2018. Disponível em: ⟨https://media.readthedocs.org/pdf/docsbitsharesorg/latest/docsbitsharesorg.pdf⟩.

SZABO, N. Formalizing and securing relationships on public networks. In: . [s.n.], 1997. Acesso em 17-11-2018. Disponível em: ⟨http://nakamotoinstitute.org/formalizing-securing-relationships⟩.

SZABO, N. Bit gold. In: . [s.n.], 2005. Acesso em 17-11-2018. Disponível em: ⟨https://unenumerated.blogspot.com.br/2005/12/bit-gold.html⟩.

TIPPETT, L. H. C. Random sampling numbers. In: . [S.l.: s.n.], 1927.

TRUEFLIP. Verifying trueflip winning combinations. In: . [s.n.], 2017. Disponível em: ⟨https://trueflipoverview.blogspot.com/2017/04/verifying-trueflip-winning-combinations.html⟩.

TSCHORSCH, F.; SCHEUERMANN, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. In: . [S.l.]: Humboldt University of Berlin, 2015.

UNIKRN. Unikoin gold: A decentralized esports gaming token. In: . [s.n.], 2017. Acesso em 17-11-2018. Disponível em: ⟨https://static.unikrn.com/1337/unikrn_bm/doc/whitepaper_en.pdf⟩.

VOORHEES, E. Satoshidice.com - the world's most popular bitcoin game. In: . [s.n.], 2012. Acesso em 17-11-2018. Disponível em: ⟨https://bitcointalk.org/index.php?topic=77870.0⟩.

WAX. How ethereum nearly killed vgo, the world's most popular dapp. In: . [s.n.], 2018. Acesso em 17-11-2018. Disponível em: ⟨https://medium.com/wax-io/how-ethereum-nearly-killed-vgo-the-worlds-most-popular-dapp-1e999e3044cb⟩.

ZIMMERMANN, P. The official pgp user's guide. In: . [S.l.]: MIT Press, 1995.