Um estudo sobre Blockchain

Igor F. Miranda
Engenharia de Computação
Universidade de Brasília
Email: igormiranda5@gmail.com

***Resumo*—The abstract goes here.**

1. Introduction

This demo file is intended to serve as a “starter file” for IEEE Computer Society conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds
August 26, 2015

2. O Bitcoin

A proposta do Bitcoin(฿) surgiu em meados de 2008 em um artigo intitulado "Bitcoin: A Peer-to-Peer Electronic Cash System"escrito por um autor sob o pseudônimo de Satoshi Nakamoto. Ele utilizou varias propostas apresentadas no b-money, Bit Gold e Hashcash para criar uma moeda digital *peer-to-peer*(P2P) que não depende de uma autoridade central, ou seja, para efetuar uma transação não é necessário a existência de uma autoridade central confiável para valida-la [2].

A grande inovação proposta por Nakamoto em seu artigo foi utilizar o conceito de prova de trabalho (*proof-of-work*) para criar um consenso distribuído confiável e resolver o problema de *Double Spending*. Tal solução pode ser utilizada para alcançar um consenso em redes decentralizadas para provar a honestidade de eleições, registros, contratos e muito mais.

Para utilizar a moeda é necessário participar da rede Bitcoin e para fazer isso deve-se possuir um cliente Bitcoin. As *Bitcoin Wallets* (Carteiras Bitcoin) são os clientes mais conhecidos para participar desse sistema onde pode-se enviar, receber e "armazenar" suas moedas. Existem vários tipos e implementações de Carteiras Bitcoins, porém, a mais conhecida é o Bitcoin Core que foi derivado da implementação original de Satoshi [1].

Atualmente exitem vários tipos de carteiras Bitcoin com diferentes níveis de segurança e propósitos. Elas são classificadas de acordo com o local de armazenamento das moedas e são divididas em cinco categorias, *Desktop wallet*, *Mobile wallet*, *Web wallet*, *Hardware wallet* e *Paper wallet* [1].

As carteiras também são classificadas de acordo com a sua autonomia e o tipo de interação com a rede Bitcoin:

- *Full node*: armazena todo o histórico de transação da rede bitcoin (Blockchain), gerencia a carteira do usuário localmente e pode iniciar uma transação diretamente com a rede Bitcoin. Ele consegue validar a Blockchain oferecendo completa autonomia e uma validação de transações independente, porém ele consome uma grande quantidade de espaço em disco.
- *Lightweight Client*: se conecta a um full node para ter acesso as transações da rede Bitcoin. Gerencia a carteira do usuário localmente, cria, valida e transmite as transações.
- *Third-party API client*: o usuário ira interagir com a rede Bitcoin através de uma API fornecida por um servidor. A carteira poderá ser armazenada com o próprio usuário ou no servidor, porém as transações são sempre gerenciadas pelo servidor.

Cada carteira Bitcoin possui uma par de chave pública/privada. A chave privada é tudo que o usuário necessita para controlar os fundos associados ao endereço da carteira Bitcoin e para comprovar a posse dos fundos usados em uma transação. A partir da chave publica utilizando uma função hash é gerado um endereço para a carteira Bitcoin. Esse par de chaves é essencial para fazer uma transação na rede Bitcoin.

2.1. Rede Bitcoin

Como a finalidade do Bitcoin é ser um sistema monetário descentralizado, justo, sem fronteiras e que não depende de uma autoridade central dizendo aos seus usuários como deve-se usa-lo, onde usa-lo, o sistema que o rege deve seguir os mesmo princípios.

Para conseguir alcançar tal objetivo a rede Bitcoin utiliza uma arquitetura chamada de peer-to-peer (P2P). Nesse tipo de arquitetura todos os nós da rede realizam as funções tanto de cliente quanto de servidor, sendo iguais hierarquicamente. Isso faz com que uma rede p2p seja descentralizada, pois não existe nenhum servidor central, serviço centralizado ou hierarquia na rede, e escalável.

O termo rede Bitcoin refere-se ao nós que participam da rede P2P Bitcoin. Porém, além do protocolo P2P a rede Bitcoin utiliza outros protocolos como o Stratum, que é utilizado por pools de mineração e carteiras leves ou

móveis, essas redes que utilizam um protocolo diferente do P2P Bitcoin são chamadas de redes Bitcoin estendidas. Essa redes estendidas comunicam-se com a rede Bitcoin através de servidores de roteamento de gateway que podem comunicar-se tanto com o protocolo P2P Bitcoin quanto o protocolo da rede estendida.

Tipos de nós

Existem quatro funcionalidades que um nó pode ter na rede: carteira, minerador, blockchain completo e roteamento. Além dessas quatro funcionalidades existe um funcionalidade para os nós da rede estendida que é de e roteamento pool, esse funcionalidade é responsável pela comunicam do nó com os servidores de roteamento de gateway (Servidores Pool) que se comunicam com a rede Bitcoin. Embora os nós em uma rede P2P sejam iguais hierarquicamente isso não significa que eles necessitam possuir as mesmas funcionalidades. Todos os nós da rede Bitcoin devem possuem a funcionalidade de roteamento e podem incluir outras funcionalidades. Os nós da rede Bitcoin são classificados da seguinte forma:

- Reference client: inclui as funcionalidades de minerador, carteira, blockchain completo e roteamento;
- Full node: inclui as funcionalidades de blockchain completo e roteamento;
- Minerador solo: inclui as funcionalidades de minerador, blockchain completo e roteamento;
- Carteira leve (SPV): inclui as funcionalidades de carteira e roteamento;
- Nós mineradores: inclui as funcionalidades de minerador e roteamento pool;
- Carteira leve(SPV) Stratum:: inclui as funcionalidades de carteira e roteamento pool Stratum.

Os nós que possuem a funcionalidade de blockchain completo mantém uma cópia completa e atualizada da blockchian, fazendo com que eles possam verificar de maneira autônoma e autoritária qualquer transação sem qualquer referência externa. Alguns nós mantêm apenas uma parte da blockchain verificando as transações utilizando um método chamado verificação de pagamento simplificado (SPV). Esses nós são conhecidos como SPV ou peso-leve já que não armazenam toda a blockchain.

Os nós com funcionalidade de minerar competem pela criação de novos blocos no Blockchain resolvendo algoritmos de prova de trabalho. Alguns nós de mineração são nós completos que matem uma cópia da Blockchian, enquanto outros são nós peso leve que participam de um pool de mineração.

As carteiras Bitcoin também podem fazer parte de um Full node para validar todas as transações da rede. Porém, cada vez mais is usuários utilizam carteiras SVP, isso vem ocorrendo porque cada vez mais os usuários vem usando carteiras em dispositivos com poucos recursos com um smartphone.

Conectando-se a rede

Para participar da rede um nó deve descobrir e conectar-se a outros nós da rede Bitcoin. Como o novo nó ainda não conhece nenhum nó da rede ele deve solicitar aos servidores DNS (DNS seeds) da rede o IP de pelo menos um nó da rede, a partir do qual pode estabelecer novas conexões.

Assim que uma ou mais conexões são estabelecidas, o novo nó deve enviar mensagens addr contendo seu endereço IP para seus vizinhos que irão retransmitir a mensagem addr recebi para seus vizinhos. Esse processo da a garantia que os novos nós serão bem conhecidos e melhor conectados. Os novos nós conectados também podem enviar mensagens getaddr para seus vizinhos, solicitando-os que retornem uma lista dos nós conectados a eles. Dessa maneira, um nó pode encontrar novos pontos para conectar-se e divulgar sua existência na rede para que outros nós possam encontra-lo.

Como os nós podem ficar offline line a qualquer momento os nós ativos precisam continuar procurando novos nós a medida que eles perdem conexões antigas.

Verificação simplificada de pagamento

2.2. Transações na rede Bitcoin

Uma transação Bitcoin informa a rede que uma carteira que controla uma quantia de Bitcoin autorizou a transferência de alguns de seus Bitcoins para outra carteira.

Cada transação contém um ou mais *outputs*, que corresponde aos Bitcoins que serão transferidos a nova carteira, e um ou mais *inputs*, que são quantias de Bitcoin do emissor. E é a partir dos *outputs* da transação serão gerados os *output* de transação não gastos que são utilizados no input de uma transação. Olhado de uma maneira mais técnica podemos falar que os *inputs* são UTXOs consumidos em uma transação enquanto os *outputs* são os UTXOs gerados em uma transação.

A única exceção para essa regra de cadeia de *inputs* e *output* é para um tipo especial de transação chamada *coinbase*. Essa transação é inserida no bloco minerado pelo minerador "vencedor", essa transação cria novos bitcoins que serão pagos para o minerador como recompensa por ter minerado o bloco. Como essa novas moedas acabaram de ser criadas essa transação não possui um *input*.

Um *output* de transação é composto por dois campos importantes: (1)uma quantia de Bitcoin em uma unidade chamada Satoshi. Essa unidade corresponde a maior divisão que pode-se fazer em uma unidade de Bitcoin que é de 8 casas decimais, ou seja, 1 Satoshi é equivalente 0.00000001฿. (2) O segundo campo é chamado de *scriptPubKey*, ele contém os requisitos necessários para utilizar o *output* de transação não gasto que será gerado.

O *input* de transação é composto por cinco campos: (1) O primeiro campo é o *Transaction Hash*, ele corresponde a referência para transação na Blockchain que contém o UOTX a ser transferido. (2)O segundo é o *output index* que identifica qual UTXO da transação será usado (primeiro é o

0). (3)Terceiro campo é o *ScriptSig* que satisfaz as condições impostas pelo *scriptPubKey*.

Os *outputs* de transação não-gastos, ou UTXOs(*unspent transaction outputs*), são *outputs* de uma transação que estão disponíveis para uma transação futura. Um UTXO é pedaço indivisível de um Bitcoin vinculado a uma chave primaria registrado na *Blockchain* e reconhecido como uma unidade de moeda na rede. Cada UTXO tem um valor como múltiplo de uma unidade chamada Satoshi. Essa unidade é a maior divisão que pode-se fazer em uma unidade de Bitcoin que é de 8 casas decimais, ou seja, 1 Satoshi é equivalente 0.00000001฿.

Ao criar um UTXO ele se torna indivisível, ou seja, seu valor não pode ser divido assim como não se pode dividir uma moeda. Se ao fazer uma transação o valor do UTXO, ou soma dos valores de vários UTXO, for maior que o valor desejado na transação é preciso gerar um troco, por consequência disso a maioria das transações Bitcoin irão gerar troco. Por exemplo,um usuário possui 3 UTXOs de Bitcoin e deseja fazer uma transação de 1 Bitcoin, essa transação precisará consumir todos os 3 UTXOs e irá produzir 2 outpus: uma para o destinatário da transação no valor de 1฿ e outro no valor de 2฿ como troco de volta para o usuário.

Essa administração dos UTXOs gastáveis em uma transação é feita automaticamente pela carteira Bitcoin do usuário, assim como sua quantia de Bitcoins. Essa quantia que toda carteira diz possuir não passa de todos os UTXOs associados a chave primaria daquela carteira que estão espalhados na Blockchain. Como efeito disso, não existe um armazenamento de saldo em uma carteira Bitcoin, o que existe na verdade são UTXOs dispersos na Blockchain vinculados a uma chave. O conceito de saldo de uma carteira Bitcoin não passa de uma abstração da operação de busca na Blockchain e a soma de todas as UTXOs pertencentes a chave primaria de uma carteira.

Os UTXOs são monitorados e armazenados na memória RAM por todos os nós da rede Bitcoin como um conjunto de dados chamado de *pool* UTXO ou UTXO *set*.

2.3. Validando transações

As transações Bitcoin são validadas através de uma linguagem script, são utilizados dois tipos de scripts para a validação: um script de travamento e um de destravamento.

Um script de travamento é uma imposição inserida em um output especificando as condições necessárias para se gastar um UTXO no futuro. O script de travamento é conhecido como *scriptPubKey* pois ele geralmente continha uma chave publica ou endereço Bitcoin, porém atualmente utiliza-se uma gama muito maior de possibilidades de script.

Um script de destravamento satisfaz as condições que forem colocas em um UTXO por um script de travamento, permitindo que o UTXO seja gasto. Eles fazem parte de todos os inputs de transação e na maioria das vezes contém uma assinatura digital produzida por uma carteira a partir de sua chave primaria, por isso é conhecido como *scriptSig*, mas nem todos os scripts de destravamento contém uma assinatura digital.

2.3.1. Linguagem Script. A linguagem script das transações Bitcoin utiliza a notação polonesa invertida, baseada em pilha(*stack*). Uma pilha possui apenas duas operações, empilhar(push) e desempilhar(pop). A operação push insere um item no topo da pilha enquanto a operação pop retira um item do topo da pilha.

Ela não é uma linguagem Turing Completa, o que significa que não se pode criar loops ou algo que de alguma maneira poderia criar um ataque de negação de serviço na rede. Outra forte característica é que ela não exige um monitoramento de estado, ou seja, não existe um estado salvo após a execução do script, toda a informação necessária para executar o script está contida no script. Isso significa que uma transação valida será valida para toda a rede.

Essa linguagem script possui dois tipos de dados, constantes de dados(números) e operadores, e executa-os da esquerda para direita. Os operadores podem desempilhar(pop) constantes de dados, opera-las e empilha-las(push) novamente. Enquanto uma constantes de dados são sempre empilhadas(push).

Atualmente existem cinco tipos padrões de scripts de transação: *pay-to-public-key-hash*(P2PKH), *Pay-to-Public-Key*(P2PK), *multi-signature*, *Pay-to-script-hash*(P2SH) e *data output* (OP_RETURN). Todos esse script serão abordados a seguir.

Pay-to-Public-Key-Hash

O script de transação Pay-to-Public-Key-Hash(P2PKH) utiliza um script de travamento que disponibiliza um *output* para um endereço Bitcoin, que é o hash da chave pública da carteira. Esse *output* pode ser destravado ao se apresentar a chave pública e a assinatura digital criada pela chave privada correspondente.

Para melhor entender o funcionamento desse script vamos supor que Bob deseja fazer uma transacão de uma quantia de bitcoins que existe em sua carteira para a carteira de um café de sua cidade. O script de travamento do UTXO que será inserido no *input* dessa transação possui a seguinte forma.

OP_DUP	OP_HASH160	Hash da chave pública de Bob	OP_EQUALVERIFY	OP_CHECKSIG

Figura 1. Exemplo de script de travamento P2PHKH

O script de destravamento que satisfaz o script de travamento P2PKH é o seguinte.

Assinatura de Bob	Chave pública de Bob

Figura 2. Exemplo de script de destravamento P2PHKH

Ao juntarmos os scripts de travamento e destravamento temos o seguinte script de validação.

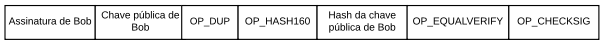

Figura 3. Exemplo de script de validação P2PHKH

Esse script de validacao será executado na seguinte forma.

1) A assinatura de Bob é inserido no topo da pilha;
2) A chave pública de Bob é inserida no topo da pilha, acima da assinatura de Bob;
3) O operador OP_DUP é executado, ele duplica o dado que está no topo da pilha e o resultado é empilhado;
4) O operador OP_HASH160 é executado, ele aplica o hash RIPEMD160(SHA256) no dado que está no topo da pilha e empilha o resultado;
5) O hash da chave pública (endereco Bitcoin) de Bob é empilhado;
6) A operaćão OP_EQUALVERIFY é executada, essa operação retira da pilha dois dados mais próximos ao topo e os compara. No caso do P2PKH será comparado o hash da chave pública de Bob empilhada em 5 com o calculada em 4. Se os hashs forem igual ambos são removidos e a operação continua;
7) É executada a operação OP_CHECKSIG, ela verifica se a assinatura de Bob combina com a chave pública fornecida, se a assinatura combinar com a chave pública fornecida a operação empilha o valor TRUE na pilha. As assinaturas no Bitcoin utilizam o Algoritmo de Assinatura Digital de Curvas Elípticas(ECDSA).

Pay-to-Public-Key

O script Pay-to-Public-Key(P2PK) utiliza a chave pública no script de travamento, ao invés do hash da chave pública como no P2PKH.

O script de travamento do P2PK possui a seguinte forma.

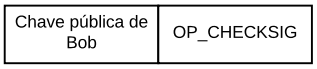

Figura 4. Exemplo de script de travamento P2PK

O script de destravamento que deve ser apresentado para se utilizar o UTXO é apenas uma assinatura a partir da chave privada de Bob.

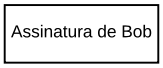

Figura 5. Exemplo de script de destravamento P2PK

Ao juntarmos os scripts de travamento e destravamento temos o seguinte script de validação.

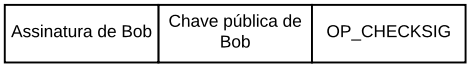

Figura 6. Exemplo de script de validação P2PK

Esse script é bem mais simples que o P2PKH, porém o seu script de travamento ocupa muito mais espaço, já que a chave pública é maior que seu hash, o que dificulta seu uso. O script P2PK é mais utilizado hoje por softwares de mineração mais antigos que não foram atualizados para utilizarem o P2PHK.

Multi-signature

Esse scrip de transação define uma condição onde N chaves públicas são registradas no script e pelo menos M dessas chaves devem ser fornecidas para a liberação de tal UTXO. Essa condição é conhecida como um esquema M-de-N, onde N é o número total de chaves e M é o número de assinaturas necessárias para a liberação do UTXO.

O script de travamento de um script Multi-signature é da seguinte forma.

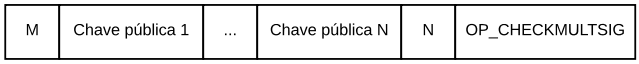

Figura 7. Exemplo de script de travamento Multi-signature

O script de destravamento Multi-signature deve apresentar M-de-N assinaturas.

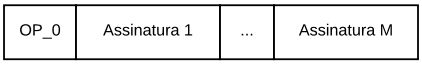

Figura 8. Exemplo de script de destravamento Multi-signature

Ao juntarmos os scripts de travamento e destravamento temos o seguinte script de validação.

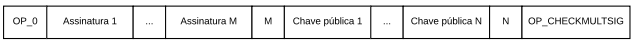

Figura 9. Exemplo de script de validação Multi-signature

Os scripts Multi-signature são os exemplos mais comuns das avançadas capacidades do script Bitcoin e são uma funcionalidade muito poderosa. Como é necessário mais de uma assinatura para liberação dos fundos, um esquema de múltiplas assinaturas como esse oferece um controle de governança corporativa e protege os fundos contra roubo, desvios ou perdas.

Apesar de todas as vantagens dos scripts Multi-signature suas desvantagens acabam inviabilizando seu uso. As transações Multi-signature são bem maiores que as transações de pagamento simples, pois elas contém chaves públicas muito longas. O fardo de uma transação extra-grande carregado pelo sistema teria que ser compensado com uma taxa de transação bem maior que as pagas pelas transações de menor tamanho, e além disso esses scripts extra-grandes teriam

que ser carregados no *pool* UTXO de cada nó da rede até que seja gasto. Outro grande problema e talvez o maior é que para cada transação desse tipo teriamos um script de transação customizado, logo teríamos que ter uma carteira que possibilita a criação de tais scripts customizados e cada usuário teria que entender como criar transações com scripts customizados. Para resolver o problema dessa dificuldades práticas e fazer a utilização de scripts complexos tão fácil quanto o P2PKH foi desenvolvido o script Pay-to-Script-Hash(P2SH).

Pay-to-Script-Hash

Nos scripts Pay-to-Script-Hash(P2SH) é utilizado o hash criptográfico do script de travamento apresentado em 2.3.1, ou seja, o script de travamento complexo e substituído pela sua impressão digital. Esse hash criptográfico codificado na Base58Check corresponde ao endereço P2PH, assim como o hash da chave pública corresponde ao endereço Bitcoin. Com o endereco P2PH resolvemos todos os proplemas apresentados em 2.3.1. Agora basta fornecer o endereco P2SH para conseguir efetuar uma transacao, um processo simples, similar ao executado no script P2PKH.

Nas transações P2SH o script de travamento é o endereço P2PH e o script de travamento do Multi-signature aqui é conhecido como script de resgate(*redeem script*), pois ele deve ser apresentado na hora do resgate dos UTXOs. O script de destravamento deve conter as assinaturas necessárias para desbloquear os UTXOs, a condição M-de-N também devem ser satisfeita no P2SH. As Tabelas 1 e 2 apresentam os scripts no esquema de Mult-signature sem o P2SH e com o P2SH respectivamente.

Tabela 1. MULTI-SIGNATURE SEM P2SH

Script de travamento	M PK1 ... PKn N CHECKMULTISIG
Script de destravamento	Ass1 ... Assn

Tabela 2. MULTI-SIGNATURE COM P2SH

Script de resgate	M PK1 ... PKn N CHECKMULTISIG
Script de travamento	OP_HASH160 <endereço P2SH> OP_EQUALVERIFY
Script de destravamento	Ass1 ... Assn

Data Recording Output (OP_RETURN)

Todas as transacões da rede Bitcoin são armazenas em um "banco de dados"distribuído chamado Blockchain, o funcionamento desse sistema será explicado mais adiante. Esse sistema potencial muito mais amplo do que apenas pagamentos, muitos desenvolvedores tentaram utilizar da segurança e resiliência da Blockchain para aplicações como cartórios digitais, certificados de ações e contratos *smart*. As primeiras tentativos surgiram ao tentar criar *outputs* de transação que registravam dados na Blockchain.

Quando a ideia de utilizar a Blockchain do Bitcoin consideram esse uso abusivo, enquanto outros acreditaram que isso mostrava a enorme capacidade dessa tecnologia. Os que eram contra argumentavam que isso causaria um "inchaço na blockchain", trazendo uma carga par os nós que armazenam a Blockchain em disco. Além disso, esse processo criaria falsas transações já que utilizam o campo de endereço do destinatário para armazenar dados, fazendo com que o UTXO jamais seja gasto. Essas transações que não podem ser gastas jamais seriam removidas do *pool* UTXO fazendo-o crescer infinitamente.

Para resolver esse problema na versão BitcoinCore 0.9.0 foi introduzido o operador OP_return, esse operador permitia que fosse adicionado 40 bytes de não-pagamento em um *output* de transação, na versão BitconCore 0.12.0 esse limite foi aumentado para 83 bytes. Esse operador cria *outputs* que são comprovadamente não gastáveis que não precisam ser registrados no *pool* UTXO.

2.4. Blockchain

O Blockchain funciona como uma de registros distribuido e compartilhado. No Boitcoin o Blockchain funciona como um livro-razão onde são armazenadas todas as transações já feitas.

Podemos abstrair o Blockchain como um empilhamento de blocos de dados, com cada novo bloco gerado sendo empilhado sobre o bloco mais recente do Blockchain. Essa visualização de blocos empilhados possibilita a utilização de termos como "altura", para se referir a posição de um bloco em relação ao primeiro bloco do Blockchain (Bloco gênese), e "topo", para se referir ao bloco mais recente do Blockchain.

Esse abstração de empilhamento de blocos é feita porque cada bloco possui a referência para o bloco anterior a ele, chamado de bloco pai. Esse referência é através de um campo no cabeçalho dos blocos que possui o identificador do bloco pai, esse identificar corresponde ao hash SHA256 do bloco. O único bloco que não possui a referência para seu bloco pai é o bloco gênese, pelo fato de ser o primeiro bloco do Blockchain.

Como cada bloco possui a o hash de seu bloco pai em seu cabeçalho isso irá influenciar no seu hash, assim, se ocorrer uma mudança em um bloco isso afetará seu hash que também afetará os blocos acima causando um efeito dominó. Esse efeito dominó garante a que um bloco que tenha algumas gerações o sucedendo não possa ser alterando, a não ser que haja um calculo forçado do hash de todos os blocos subsequentes. Essa é uma das características chave para a segurança do Blockchain, pois a mudança de um bloco que já possui uma certa profundidade exigira um processo computacional enorme.

Estrutura de um bloco

3. Conclusion

The conclusion goes here.

Acknowledgments

The authors would like to thank...

Referências

[1] A. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, 2017.

[2] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Journal for General Philosophy of Science*, 39(1):53–67, 2008.