DISSERTAÇÃO DE MESTRADO

PROPOSIÇÃO DE UM MODELO DE BLOCKCHAIN PARA OPERAÇÕES INTERBANCÁRIAS

Raphael Alves Bruce

Brasília, dezembro de 2022

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

PROPOSIÇÃO DE UM MODELO DE BLOCKCHAIN PARA OPERAÇÕES INTERBANCÁRIAS

Raphael Alves Bruce

Dissertação de Mestrado submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Daniel Guerreiro e Silva, Ph.D., FT/UnB _____
Presidente

Prof. Ugo Silva Dias, Ph.D., FT/UnB _____
Suplente

Prof. Georges Daniel Amvame Nze, Ph.D., FT/UnB _____
Membro Externo ao Programa

Prof. Daniel Alves da Silva, Ph.D., IBMEC _____
Membro Externo

FICHA CATALOGRÁFICA

BRUCE, RAPHAEL ALVES
PROPOSIÇÃO DE UM MODELO DE BLOCKCHAIN PARA OPERAÇÕES INTERBANCÁRIAS [Distrito Federal] 2022.
xvi, 95 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2022).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Blockchain de Consórcio	2. Operações Interbancárias
3. Escalabilidade	4. Desempenho
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

BRUCE, R. A. (2022). *PROPOSIÇÃO DE UM MODELO DE BLOCKCHAIN PARA OPERAÇÕES INTERBANCÁRIAS*. Dissertação de Mestrado, Publicação PPGEE.DM 800/22, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 95 p.

CESSÃO DE DIREITOS

Raphael Alves Bruce
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

AGRADECIMENTOS

Agradeço ao meu orientador, professor Dr. Rafael Timóteo de Sousa Júnior, pela oportunidade de desenvolvimento deste trabalho no PPGEE Programa de Pós-graduação em Engenharia Elétrica. E ao professor Dr. Robson de Oliveira Albuquerque pela disponibilidade em orientar e pelas valiosas dicas e opiniões relacionadas à esta dissertação.

Aos professores, Fábio Mendonça e Georges Amvame, pelas dicas e ajudas essenciais para o desenvolvimento deste trabalho. Agradeço também aos membros da banca pela oportunidade.

À minha esposa Jéssica por incentivar meu esforço e manter a paciência durante os momentos em que meu foco foi este trabalho.

À minha mãe Heliana e ao meu irmão Daniel pelo apoio.

Por fim, agradeço à Deus.

RESUMO

PROPOSIÇÃO DE UM MODELO DE *BLOCKCHAIN* PARA OPERAÇÕES INTERBANCÁRIAS

Autor: Raphael Alves Bruce
Orientador: Rafael Timóteo de Sousa Júnior
Coorientador: Robson de Oliveira Albuquerque
Programa de Pós-graduação em Engenharia Elétrica - PPGEE
Brasília, Dezembro de 2022

A tecnologia *blockchain* surgiu como o principal mecanismo das criptomoedas e por manter um banco de dados distribuído que depende de processos de consenso para a persistência dos dados passou a ter seu uso considerado em várias outras aplicações, principalmente devido às suas características de disponibilidade, segurança e descentralização.

O momento para a tecnologia também se mostra bastante favorável devido aos cenários atuais em que negocialmente se exige cada vez mais integração e compartilhamento de dados entre parceiros de negócios. Como a rede *peer-to-peer blockchain* provê um *ledger* distribuído entre os nós participantes compartilhado com transparência, imutabilidade e confiabilidade, logo os investimentos nesta aumentaram, credenciando-a como solução tecnológica.

Considerando esta abordagem, esta dissertação propõe um modelo de uso de *blockchain* de consórcio para comunicação de operações interbancárias entre entidades financeiras parceiras, substituindo interfaces legadas baseadas no processamento de arquivos em lotes. Este modelo tem preocupação com escalabilidade e desempenho e possui como principais contribuições: a) a utilização do *Patricia Merkle Tree* para estabelecer um processo de encadeamento dos blocos na cadeia que permita melhor escalabilidade quando comparado com às soluções mais disseminadas em *blockchain*, b) a proposição de um mecanismo de consenso híbrido, que mantenha as vantagens e supere os problemas do prova de autoridade e da tolerência às falhas bizantinas práticas (PBFT) principalmente, tendo desempenho superior ao mecanismo de prova de trabalho, geralmente implementado nas *blockchains*, e c) a adoção de diretrizes para privacidade de dados adicional para manter o sigilo das informações em um contexto de transparência e distribuição, atendendo a exigência de que todos os dados pessoais e informações sensíveis permaneçam confidenciais. E camada de auditoria para manter as regras acordadas pelo consórcio com a utilização de *smart contracts* para monitoramento automático e permanente.

A proposta é validada com o desenvolvimento e testes de protótipos correspondentes, considerando a avaliação de funcionalidades propostas e definidas para o modelo, bem como o desempenho em relação aos mecanismos geralmente escolhidos nas implementações mais comuns em *blockchain*.

Palavras-chave: ***blockchain* de consórcio; operações interbancárias; escalabilidade; desempenho; privacidade; auditoria; *smart contracts***.

ABSTRACT

PROPOSAL OF A BLOCKCHAIN MODEL FOR INTERBANKING TRANSACTIONS

Author: Raphael Alves Bruce
Supervisor: Rafael Timóteo de Sousa Júnior
Co-Supervisor: Robson de Oliveira Albuquerque
Post-graduate Program on Electrical Engineering - PPGEE
Brasilia, December of 2022

Blockchain technology has emerged as the primary mechanism of cryptocurrencies and by maintaining a distributed database that relies on consensus processes for data persistence it is considered for use in many other applications, mainly due to its characteristics of availability, security and decentralization.

The timing for technology is also very favorable due to current scenarios where business integration and data sharing between business partners is increasingly required. As the peer-to-peer blockchain network provides a ledger distributed among the participating nodes that is shared with transparency, immutability and reliability, investments in it have increased, crediting it as a technological solution.

Considering this approach, this dissertation proposes a model for using consortium blockchain to communicate interbank transactions between partner financial entities, replacing legacy interfaces based on batch file processing. This model is concerned with scalability and performance and has as main contributions: a) the use of Patricia Merkle Tree to establish a chaining process that allows better scalability when compared to the most widespread blockchain solutions, b) the proposition of a hybrid consensus engine that retains the advantages and overcomes the problems of proof of authority and practical byzantine fault tolerance (PBFT) primarily, outperforms the proof of work mechanism, generally implemented in blockchains, and c) adoption of additional data privacy guidelines to maintain the confidentiality of information in a context of transparency and distribution, meeting the requirement that all personal data and sensitive information remains confidential. Establishing an audit layer to maintain the rules agreed upon by the consortium with the use of smart contracts for automatic and permanent monitoring.

The proposal is validated by the development and testing of its corresponding prototype, considering the evaluation of the proposed and defined functionalities for the model, as well as the performance in comparison to the mechanisms generally chosen in the most common blockchain implementations.

Keywords: **consortium blockchain; interbank transaction; scalability; performance; privacy; audit; smart contracts**.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

PBFT	*Practical Byzantine Fault Tolerance*
P2P	*Peer-to-Peer*
BaaS	*Blockchain as a Service*
PoW	*Proof of Work*
PoA	*Proof of Authority*
PoS	*Proof of Stake*
JSON	*JavaScript Object Notation*
BFT	*Byzantine Fault Tolerance*
SHA256	*Secure Hash Algorithm 256-Bit*
PKI	*Public Key Infrastructure*
CA	*Certification Authority*
API	*Application Programming Interface*
CTA	*Computacional Threshold Adversary*
TAM	*Threshold Adversary Model*
DAO	*Decentralized Autonomous Organization*
VPN	*Virtual Private Network*
DDoS	*Distributed Denial of Service Attack*
LGPD	Lei Geral de Proteção de Dados
IDE	*Integrated Development Environment*
IDLE	*Integrated Development and Learning Environment*
IaaS	*Infrastructure as a Service*
RAM	*Random Access Memory*
RAM	*Central Processing Unit*
VS	*Virtual Studio*
GUI	*Graphical User Interface*
SHA-3	*Secure Hash Algorithm version 3*

1 INTRODUÇÃO

A tecnologia *blockchain* vem sendo pesquisada e debatida com relação às possíveis aplicações nos ambientes corporativos e como solução para cenários em que haja problemas de confiança nas interoperações entre diferentes entidades. Apesar de sua aplicabilidade ainda não estar bem delimitada como solução tecnológica, existem diversas abordagens sendo fomentadas atualmente, investimentos consideráveis sendo aplicados e bastante expectativa no uso do *blockchain* para várias finalidades. Neste contexto, deve haver cautela na definição do cenário de uso do *blockchain*, pois, considerando apenas o interesse do mercado com a tecnologia, muitas empresas empreenderam projetos de *blockchain* pela implantação da tecnologia por si só, em vez de resolver problemas de negócios [3].

O interesse é alto e há potencial nas tecnologias *blockchains*, porém um fator de sucesso para sua aplicação é o alinhamento às necessidades de negócios específicas. Bem como, conforme aumento da maturidade dos estudos com relação a tecnologia, vários problemas inerentes passaram a ser expostos e explorados, aumentando as customizações e as soluções propostas para superação de questões relacionadas à escalabilidade, desempenho, privacidade e segurança.

As pesquisas sobre *blockchain* estão alinhadas à esta tendência, assim até 2017 a maioria explorava o funcionamento da tecnologia, as plataformas de criptomoedas e as possibilidades de aplicações, como em [4], [5] e [6]. Porém, publicações mais recentes estão voltadas para as implicações dos mecanismos da tecnologia, as soluções específicas aos problemas da *blockchain* e as necessidades de uso e aplicações da mesma, como em [7], [8] e [9]. Nota-se uma preocupação maior com relação às aplicações práticas da tecnologia e seus desdobramentos, demonstrando maior maturidade na compreensão sobre *blockchain* e suas capacidades [3].

Atualmente, os principais desafios são a continuidade do processo de amadurecimento da tecnologia com a superação dos seus problemas inerentes e a correta avaliação do contexto de aplicação da tecnologia. Assim, geralmente, o *blockchain* é uma alternativa em modelos que entidades não confiam entre si e querem interagir e mudar o estado de um sistema, sem estarem dispostas a ter um intermediário sempre disponível para validar as transações.

De maneira mais específica, as principais motivações para utilização da tecnologia *blockchain* envolvem a avaliação dos seguintes pontos: a) a necessidade de persistir dados de forma organizada, b) o interesse de compartilhar esses dados, c) a intenção de não depender de uma entidade validadora central, e d) a ausência de confiança total entre as entidades que compartilham os dados. Para ilustrar este fluxo de decisão com relação ao uso do *blockchain* como solução tecnológica, temos o fluxograma 1.1 abaixo.

Contudo, duas questões são iminentes para o uso de *blockchain*: a) compartilhar dados entre diferentes entidades e b) ausência de confiança entre as partes envolvidas. Resolver essas situações é vantajoso para os negócios atualmente pela economia de gastos, diminuição dos riscos pela transparência e auditoria, e aumento da conformidade com a verificação imediata das transações. As superações aos problemas inerentes da tecnologia são conseguidas pelas escolhas dos mecanismos que irão compor o modelo de uso de *blockchain* a ser estabelecido, que irá depender do problema de negócio que a tecnologia solucionará.

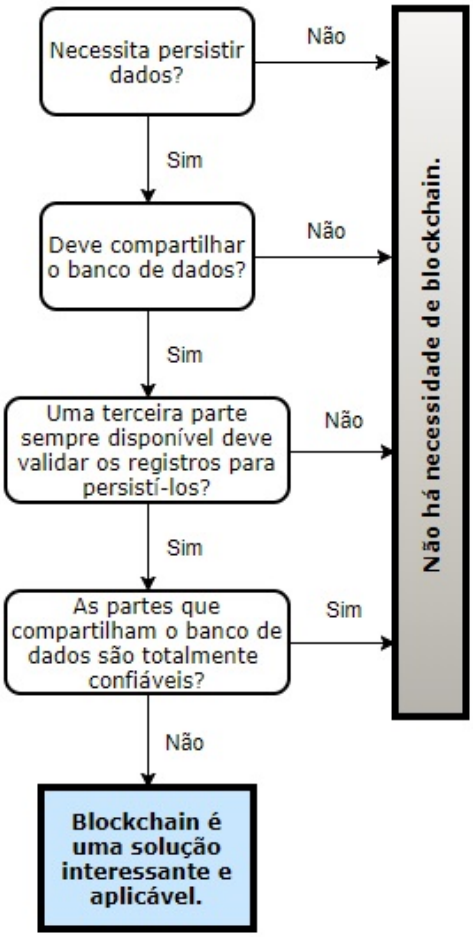

Figura 1.1: Fluxograma para decisão sobre o uso do *blockchain*

1.1 MOTIVAÇÕES E JUSTIFICATIVAS

O cenário das interface de sistemas entre empresas, entidades financeiras e órgãos reguladores para execução de um fluxo de negócio determinado é corriqueiro, pois nos dias atuais cada vez mais empresas compartilham informações e negócios. Assim, há urgência para que estas cooperarem entre si através de soluções de alta disponibilidade, garantindo imutabilidade dos dados compartilhados, escalabilidade, desempenho, privacidade dos dados pessoais, auditoria e rastreabilidade [10].

Considerando este contexto, a tecnologia *blockchain* é uma alternativa às soluções tradicionais de comunicação de operações interbancárias, em que arquivos de dados são transmitidos entre os servidores das entidades que mantém o fluxo de negócio para que cada nó faça seu processamento centralizado e transmita os retornos, na maioria das vezes assíncronos e demorados. Assim, a utilização da tecnologia *blockchain* promove um novo cenário em que as instituições possam compartilhar bancos de dados distribuídos, que mantenham as informações do fluxo transacional de determinados produtos, mesmo que não haja confiança total entre as partes, de forma que as transações sejam verificadas pela solução compartilhada com transparência e sem a necessidade de validação por uma autoridade reguladora, a qual, caso exista, pode ser apenas comunicada sobre as operações ou acessar a *blockchain* para auditoria e monitoramento.

Assim, diminui-se os pontos de possibilidade de falha nas operações interbancárias e gera-se economia pela redução do desenvolvimento de soluções para processar as transações em cada nó das interfaces e pela possível dispensa ou renegociação dos termos de atuação da entidade verificadora, que não terá mais a responsabilidade exclusiva de promover a confiança das operações realizadas.

A utilização de uma arquitetura distribuída para comunicação de operações interbancárias com *blockchain* apresenta mecanismos para escalabilidade, desempenho, segurança e privacidade, obtendo consequentemente vantagens sobre os processos tradicionais das operações interbancária baseados nas transferências de arquivos, além de conseguir garantir transparência, autonomia no monitoramento e verificação, disponibilidade, rastreabilidade e colaboração.

Consequentemente, a utilização da *blockchain* para operações interbancárias reflete em redução de custos pelo compartilhamento da solução, diminuição do trabalho de *backoffice* exigido pelas soluções legadas, que precisam de vários processos de conformidade no acompanhamento das operações interbancárias e seus reflexos, e decremento dos esforços com operação para execução e acompanhamento das *schedules* de processamento em lotes e dos mecanismos de transmissão dos arquivos.

1.2 OBJETIVOS

Este trabalho possui o objetivo de estabelecer um modelo de uso de *blockchain* para comunicação de operações interbancárias entre entidades financeiras com alta disponibilidade, baixa latência, auditoria constante e sem a necessidade de validação por órgão terceiro. Possibilitando a substituição das interfaces legadas baseadas no processamento em lote de transações enviadas por arquivos para efetivação das operações interbancárias.

O embasamento do trabalho prescinde de estudos sobre arquitetura e mecanismos de *blockchain*, seus modelos de governança, suas características de consenso e segurança, as questões de confiança em sistemas distribuídos, bem como, um levantamento dos problemas das soluções atuais e os mecanismos do modelo proposto que os contornem.

No sentido de alcançar os objetivos desta dissertação, a pesquisa constitui-se de identificação e caracterização da tecnologia *blockchain*, apresentando questões relacionadas a suas aplicações, modelos de referência e o relacionamento com a problemática da confiança em sistemas distribuídos.

Seguindo com a compreensão do problema das soluções atuais de comunicação de transações interbancárias, estabelecendo o modelo de governança mais adequado para o contexto e os mecanismos e as implementações inerentes à esta definição. Finalizando com o desenvolvimento de um protótipo de *software*, que permita a avaliação da proposta em cenários de teste das funcionalidades do *blockchain* implementado.

Assim, com a intenção de alcançar o objetivo geral desta proposta, os seguintes objetivos específicos foram definidos:

- Estudar funcionalidades dos modelos *blockchain*.
- Propor modelo de *blockchain* para operações interbancárias.

- Especificar o funcionamento do modelo *blockchain* proposto.
- Apresentar um protótipo com requisitos definidos implementados.
- Realizar validação da proposta por intermédio de testes para verificação das funcionalidades da *blockchain*.

1.3 PUBLICAÇÃO VINCULADA AO TRABALHO

- [11] Bruce, Raphael A.; De Sousa, Jr., Rafael T.; Mendonca, F. L. L.; Pimentel, J. P.; Holanda, M. T. ; Filho, Francisco L. De Caldas. *Blockchain para operações interbancárias.* In: Conferência Ibero-Americana WWW/Internet 2019, 2019, Lisboa, Portugal. IADIS Press, 2019. ISBN: 978-989-8533-96-8, v. 1, p. 265-268.

1.4 ORGANIZAÇÃO DO TRABALHO

Esta dissertação é composta por cinco capítulos, incluindo esta introdução.

O capítulo 2 trata de trabalhos relacionados e das bases conceituais da dissertação, o que inclui descrição dos modelos de governança de tecnologia *blockchain*, dos mecanismos relacionados ao trabalho e de suas características, do modelo de referência estabelecido pelo *Bitcoin*, dos *smart contracts*, e sobre a questão da confiança em sistemas distribuídos e auditoria.

No capítulo 3, o modelo de *blockchain* para operações interbancárias é apresentado, constituindo a contribuição central da dissertação. São apresentados os problemas das soluções atuais, as discussões sobre aplicações da tecnologia *blockchain* e acerca das relações de confiança, os mecanismos inerentes do modelo proposto e sua arquitetura de organização e funcionamento, as funcionalidades de auditoria e as vantagens da proposta.

O capítulo 4 destina-se à avaliação da proposta, descrevendo os protótipos desenvolvidos, os cenários de validação das funcionalidades, as caraterísticas relacionadas ao desenvolvimento e os pontos relevantes à discussão das validações obtidas.

O capítulo 5 apresenta a conclusão do trabalho e propõe trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA E TRABALHOS RELACIONADOS

Neste capítulo são apresentados os conceitos relacionados a *blockchain* e a confiança nos sistema distribuídos, que auxiliam na compreensão do modelo proposto no Capítulo 3 e são fundamentais para delimitação do objetivo do trabalho e de suas motivações. Além de discutir os temas vinculados às provas de conceitos e validações expostas no Capítulo 4 correspondentes ao modelo proposto, que são importantes para entendimento do funcionamento de sua arquitetura.

Como o objetivo do trabalho é propor um modelo de *blockchain* para comunicação das operações interbancárias entre entidades financeiras em substituição às interfaces legadas baseadas no processamento em lotes de arquivos, este capítulo aprofundará no tema *blockchain* e em suas aplicações, apresentando seus modelos de governança e as características principais da tecnologia. Apresentará e explorará os mecanismos da tecnologia que participam do modelo proposto, bem como os conceitos relacionados aos objetivos do trabalho e às discussões associadas. Além de abordar a confiança nos sistemas distribuídos, considerando a *blockchain* e as soluções bancárias, e discorrer sobre *smart contracts* e auditoria.

Para compreensão do modelo proposto neste trabalho e suas motivações, este capítulo trará discussões sobre a utilização de *blockchain* como solução tecnológica corporativa e apresentará os devidos contrapontos em relação a sua implementação mais famosa, o *Bitcoin*. Explorando as características relacionadas à perspectiva do trabalho com relação ao consenso da rede, à estrutura da cadeia de blocos com foco em seu cabeçalho e aos atributos de segurança e privacidade, com foco em autenticação e federação e à categorização de privacidade de dados incorporada e de privacidade de dados adicional.

2.1 CONCEITOS E CARACTERÍSTICAS DA *BLOCKCHAIN*

Blockchain é uma solução distribuída cujos registros são organizados em blocos vinculados como uma cadeia mantida por uma rede ponto a ponto, do inglês *peer-to-peer* (P2P), que permite vários nós trabalhando em conjunto para estabelecer uma rede unificada e descentralizada de comunicação e compartilhamento de dados, utilizando algoritmos de consenso em substituição à necessidade de uma autoridade central que torne toda a rede confiável.

Este banco de dados é conhecido como livro-razão, ou em inglês *ledger*, e mantem um inventário distribuído compartilhado numa rede *peer-to-peer* descentralizada. Para manipulação dos dados é permitida apenas as operações de leitura e gravação, não sendo possível alteração e deleção. As principais características das soluções em *blockchain* são transparência, imutabilidade e confiabilidade, uma vez que os dados armazenados não podem ser corrompidos ou alterados, os registros são mantidos de forma distribuída entre os nós da rede e a persistência dos dados depende de verificação por mecanismos de consenso.

As soluções *blockchains* têm como principal estrutura arquitetural a disponibilização de um banco de

dados distribuído e criptografado mantido em uma rede *peer-to-peer*, cuja persistência dos registros esteja condicionada à verificação por algoritmos de consenso. Assim, as principais características dessa estrutura são: a imutabilidade, a descentralização, a segurança, a distribuição, o consenso e a integração. Fatores que fomentam o uso da tecnologia visto sua compatibilidade com as necessidades dos negócios do mundo atual.

Como na *blockchain* cada nó da rede possui uma cópia da cadeia, os registros são persistidos apenas após verificação de consenso da rede e as operações permitidas são de inserção e leitura, os mecanismos da solução garantem proteção e imutabilidade. E, pela rede ser mantida de forma descentralizada, a governança do mecanismo é decidida pelas necessidades e características de negócio, invés de ser limitada por questões tecnológicas que levam à centralização por falta de recursos para superar os problemas das arquiteturas distribuídas. Sendo que a descentralização dos registros traz transparência, independência e menor propensão a falhas, ataques e corrupção.

A segurança também é aprimorada pelo perfil descentralizado da solução e pelo uso de criptografia e encadeamento por *hashes*, pois as operações de *hash* envolvidas nas estruturas de dados de identificação dos blocos são complexas e alterações ou reversões causam dissonâncias, assim há uma irreversibilidade inerente da cadeia.

A distribuição ajuda no contexto da segurança pela transparência que traz devido ao compartilhamento dos dados e pelo maior esforço necessário para atacar várias entidades descentralizadas invés de uma entidade central, além de possibilitar maior velocidade de processamento em comparação aos contextos centralizados, bem como melhor disponibilidade da solução, que não depende de uma unidade centralizada para se manter funcional.

Essas características ajudam para que as soluções *blockchains* sejam importantes aos contextos de necessidade de integração de dados, pois mantem uma estrutura distribuída, de autoridade descentralizada, com mecanismos que garantam consenso e segurança. Tendo o consenso papel fundamental no processo de tomada de decisão entre o grupo de nós ativos na rede, o que auxilia na independência da rede distribuída com relação às entidades centralizadas e contribui para segurança.

Assim, a arquitetura *blockchain* é adequada ao objetivo do trabalho de propor um modelo para comunicação de operações interbancárias entre entidades financeiras, sendo o contexto distribuído um fundamental motivador por descentralizar as informações compartilhadas de negócio, além da organização de forma transparente e sem a necessidade de validação por uma autoridade reguladora, bem como a imutabilidade dos registros. Fatores fundamentais para o negócio bancário pela sensibilidade dos dados, necessidade de integração, fragilidade nas relações de confiança entre os parceiros de negócio e possibilidade de excluir a dependência de entidades autorizadoras.

2.2 APLICAÇÕES DA TECNOLOGIA *BLOCKCHAIN*

Os principais exemplos de aplicações de *blockchain* são relacionados às transações de bens, assim temos: a) as moedas digitais, como *Bitcoin*, *Ether* e *LiteCoin*, b) os registros de bens, como *ColorCoin* e *Counterparty*, e c) as plataformas de gestão de ativos, como *Ripple*, *Hyperledger*, *Tendermint* e *Corda*.

Outro uso já bem difundido são as plataformas de desenvolvimento, como *Ethereum*, *Cosmos*, *Quorum* e *Hyperledger Fabric*. As publicações [7] e [12] apresentam quadros resumos com descrições dessas aplicações e suas principais características.

Considerando os investimentos na tecnologia é possível identificar duas principais motivações para o uso da *blockchain*, que seriam: a) manter um registro compartilhado transparente de dados ou b) retirar a necessidade de entidades autorizadoras para garantir confiança entre as operações envolvendo diversos participantes.

Assim, são exemplos de aplicações em *blockchain* atreladas à primeira motivação: gestão da cadeia de suprimentos, identidade digital, votação, *tokenização* de *assets*, cuidados com a saúde (compartilhamento de históricos médicos), arrecadação de valores e caridade, tabelionato, segurança alimentar (compartilhamento de histórico de produção e abastecimento de alimentos), propriedade intelectual (proteção de direitos autorais e de *royalty*), automóveis (compartilhamento de histórico de produção e distribuição), registro de propriedade imobiliária, internet das coisas.

E estão no rol do segundo grupo de investimento as aplicações em: criptomoedas, armazenamento distribuído em nuvem, economia colaborativa (sistema socioeconômico construído em torno da partilha de recursos humanos e físicos), música (produção e distribuição), contratos inteligentes, seguros, bancos.

A primeira grande aplicação da tecnologia foi com as criptomoedas, tendo sido o *Bitcoin* o maior exemplo e de longe a aplicação mais utilizada. Assim, inicialmente, o interesse e a pesquisa sobre *blockchain* foram focados no *Bitcoin* e nas criptomoedas, por isso este é o tipo de aplicação na qual a tecnologia demonstra mais maturidade, mas também é o ramo em que ela encontra os desafios mais debatidos até o momento sobre desempenho e regulamentação. Estas questões ainda não foram superadas e a tecnologia está distante de ser utilizada como mecanismo comum de pagamentos. Porém, conforme a pesquisa sobre *blockchain* foi avançando, novas aplicações e possibilidades de soluções atreladas foram surgindo e há uma expectativa futura promissora e abrangente [10].

Após o advento das criptomoedas, outra aplicação que deu força à tecnologia foi a organização das plataformas *blockchain*. Estas, geralmente constituídas em plataformas *opensource*, provêm funciolidades consumidas como serviços que são mantidos nas plataformas *blockchain*, em um contexto de *Blockchain* como um Serviço, do inglês *Blockchain as a Service* (BaaS). Assim, o BaaS é um serviço que permite que os clientes aproveitem a base de nuvem para construir, hospedar e usar suas próprias soluções *blockchains* de aplicativos, enquanto o provedor de serviços é responsável por gerenciar a infraestrutura e mantê-la ágil e operacional, ajudando na adoção da tecnologia pelas empresas e utilização para soluções de problemas e negócios corporativos [7].

2.2.1 *Blockchain* como Solução Tecnológica

O uso de *blockchain* nas corporações ganhou espaço e os investimentos passaram a ser mais robustos, porém os projetos continuam com dificuldades para deslanchar principalmente pelos seguintes motivos: desconhecimento sobre o propósito da tecnologia, preconceber que *blockchain* não precisa de customização, confundir as funcionalidades como uma solução completa, entender *blockchain* apenas como uma base de dados ou uma solução de armazenamento, assumir que a interoperabilidade entre *blockchains* existe na-

tivamente, preconceber que contratos inteligentes são problemas fechados e ignorar regulamentações que afetem a constituição de redes distribuídas *peer-to-peer* [3].

Complementam esta lista de dificultadores as considerações sobre os custos de implementação das soluções *blockchains*, a falta de profissionais especializados, o ambiente regulatório desconhecido e a dificuldade de migração das soluções tradicionais para as novas tecnologias. Este desconhecimento pode causar desapontamento e falhas nos projetos utilizando a tecnologia, por isso a aplicação da tecnologia nas organizações está bastante conectada à solução específica para um problema relacionado, considerando um contexto coerente para uso de *blockchain* e adotando mecanismos adequados na implementação, conforme o problema a ser atacado pela solução.

Um ponto favorável é o amadurecimento da *blockchain* como solução tecnológica conforme a evolução das pesquisas na área e o aumento dos *cases* de implementação utilizando bancos de dados distribuídos em redes *peer-to-peer*. Essa evolução permite que as organizações compreendam o contexto de aplicação da *blockchain* e como esta pode ser uma solução interessante considerando seus problemas de negócio, tendo sido a motivação desta dissertação ao estabelecer como objetivo uma proposta da arquitetura de solução utilizando *blockchain* para o problema concreto de comunicação de operações interbancárias entre entidades financeira.

2.3 CONFIANÇA EM SISTEMAS DISTRIBUÍDOS

O problema da confiança é uma questão recorrente na programação distribuída visto que vários nós são interligados para manter uma solução tecnológica baseada no compartilhamento de dados e do estado do sistema. Assim, um problema fundamental é garantir que esses nós confiem na atualização dos dados e nas respostas do sistema, ou seja, que não haja desconfiança de que alguns nós estejam prejudicando a rede.

Esta questão é urgente no contexto atual em que o ecossistema empresarial busca cada vez mais parcerias entre seus participantes, compartilhando dados e soluções tecnológicas. Nesse universo também é frequente a necessidade de soluções interoperacionais, em que a infraestrutura tecnológica de uma organização tenha capacidade de comunicação com a vertical de outra entidade. Assim, a manutenção dessas redes compartilhadas deve ser baseada em confiança, auditoria e consenso para evitar prejuízos aos participantes.

2.3.1 Considerações sobre o Problema dos Generais Bizantinos

O problema dos generais bizantinos foi introduzido por Lamport, Shostak e Pease, nos idos de 82 considerando as implicações do problema de confiança em redes distribuídas, estabelecendo que sistemas de computadores confiáveis devem lidar com componentes com defeito que fornecem informações conflitantes para diferentes partes do sistema.

Assim, o problema considera o contexto apresentado no artigo *The Byzantine Generals Problem* [13] de que um grupo de generais do exército bizantino acampados com suas tropas em uma cidade inimiga, apenas se comunicando por mensageiros, devem concordar com um plano de batalha comum. No entanto, um ou

mais deles podem ser traidores que tentarão confundir os outros. Desta forma, o problema é encontrar um algoritmo para garantir que os generais leais chegarão a um acordo.

O trabalho apresenta uma série de abordagens e premissas para atacar o problema, tais como: a) todos os generais leais decidem o mesmo plano de ação, b) um pequeno número de traidores não pode fazer com que os generais leais adotem um mau plano, c) todo general leal recebe as mesmas informações, d) se o enésimo general é leal, então o valor que ele envia deve ser usado por todos os generais, e) todos os tenentes leais obedecem à mesma ordem, e f) se o general é leal todo tenente obedece sua ordem.

Lamport, Shostak e Pease defendem a tese de que usando apenas mensagens orais o problema é solucionável somente se mais de dois terços dos generais forem leais, pois um único traidor pode confundir dois generais leais. E indicam que é a capacidade dos traidores mentir que torna o problema dos generais bizantinos tão difícil, assim essa capacidade deve ser restringida, considerando o uso de mensagens assinadas em um contexto em que a assinatura de um general leal não pode ser forjada e qualquer um pode verificar a autenticidade da assinatura de um general. Além de estabelecer que barreiras físicas habilitam algumas restrições sobre quem pode enviar mensagens para quem, assim o algoritmo deve considerar que os generais apenas enviem mensagens para quem podem.

Porém, a principal consideração do trabalho *The Byzantine Generals Problem* que o modelo de referência *blockchain* utiliza é que a confiança é atingida quando vários processos diferentes são utilizados para calcular o mesmo resultado e, em seguida, uma votação é realizada e o resultado majoritário é definido como valor único. Sendo que o uso da votação majoritária para obter confiabilidade é baseada na suposição que todos os participantes "leais" produzirão a mesma saída, sob duas condições: a) todos os processos devem utilizar a mesma entrada e b) a unidade de entrada não pode ser inconsistente.

A arquitetura *blockchain* utiliza premissas do problema dos generais bizantinos para garantir confiança no sistema distribuído, entre as quais: a) apenas uma decisão final é estabelecida, se o bloco é válido ou não, b) a rede é sincronizada no processo de consenso, c) os nós que não participam do consenso, mas estão na rede, devem aceitar o resultado, d) os nós devem ser credenciados pelo mecanismo de consenso definido para trocar as mensagens, e) operações de infraestrutura de chave pública são utilizadas, e f) em alguns *blockchains*, o mecanismo de consenso estabelece um mínimo de nós participantes.

2.3.2 Confiança pelo Consenso Distribuído do *Blockchain*

No contexto da *blockchain*, a questão de confiança está relacionada com a concordância quanto às transações válidas de um sistema distribuído, assim a exigência é que a maioria concorde na persistência de um bloco de transações na cadeia. Desta forma, a arquitetura *blockchain* estabelece como mecanismo para confiança em uma rede distribuída a utilização de mecanismos de consenso para condicionar a apropriação de transações na cadeia.

O problema de consenso pode ser afirmado em três propriedades: a) **concordância**, pois não é possível dois processos corretos decidirem por blocos diferentes, b) **validade**, já que o bloco validado é um dos blocos propostos no processo de verificação, e c) **finalização**, pois todos os processos corretos eventualmente decidem o bloco [14]. Por ser uma solução distribuída, uma *blockchain* minimamente deve apresentar um método para chegar a um acordo sobre uma cadeia compartilhada de blocos, em que cada bloco contenha

uma sequência de transações.

Na computação distribuída tradicional a suposição mais comum para tentar superar o problema de confiança é o Modelo de Limite do Adversário, do inglês *Threshold Adversary Model* (TAM). Neste modelo, normalmente existe a suposição de um limite entre dois parâmetros: a) **o número total de partes**, frequentemente indicado por n, e b) **o número total de partes** que o adversário pode controlar, geralmente indicado por f, sendo que os limites típicos definidos são a minoria $(n > 2f)$ ou um terço $(n > 3f)$. Assim, os limites indicam que não é o caso de que qualquer um seja incluído no grupo dos n participantes, ou seja, a participação depende de autorização para ser possível assumir um conjunto fixo de n participantes.

A abordagem *blockchain* adota outro modelo de modo geral principalmente devido ao mecanismo de prova de trabalho, adotado pelo principal modelo de referência que é o *Bitcoin*, que é o Limite Computacional do Adversário, do inglês *Computacional Threshold Adversary (CTA)*. Neste modelo, em vez de limitar o número total controlado pelos possíveis adversários em relação ao número total de partes, o modelo limita a quantidade total de poder computacional que o adversário possui em relação ao total computacional disponível considerando todas as partes. Assim definimos: a) **a quantidade total de energia computacional**, denotada por e, e b) **a quantidade total de poder computacional que o adversário pode controlar**, indicada por a, para trabalhar com a suposição de que o adversário controla uma minoria do poder computacional considerando o limite $e > 2a$, sendo em alguns casos suposto que o adversário controla n frações menos que a maioria, assim $e > 2(1 + n)a$. Neste contexto a participação não depende de autorização, pois qualquer um que possa resolver os problemas criptográficos pode se juntar ao sistema [15].

Considerando as abordagens, a solução *blockchain* apresenta como vantagem estabelecer uma suposição mais consistente ao utilizar como parâmetro a execução de operações matemáticas para medir o poder computacional do universo de nós participantes, além de trabalhar numa abordagem que não necessita de autorização prévia para participação. Por estes fatores, a *blockchain* passou a ser celebrada como uma solução importante para o contexto do problema de confiança dos sistemas distribuídos.

2.3.3 Soluções Bancárias e as Questões de Confiança

A questão de confiança é muito importante no contexto das soluções bancárias, pois as entidades financeiras tem urgência em avançar em soluções para compartilhamento de dados e interoperabilidade de serviços, mas não podem deixar de implementar mecanismos que garantam a manutenção do sigilo bancário e proteção das informações, tanto devido à necessidade de privacidade, quanto por vantagem competitiva.

A cobrança nessas entidades para transformação digital tornando a interação com o cliente mais fluida e rápida, além da colaboração entre os participantes para ampliar a gama de serviços e produtos ofertados, é frequente e pautada na garantia de disponibilidade, desempenho e segurança, visto as características esperadas das ferramentas tecnológicas de negócios bancários de facilidade, robustez e confiabilidade.

Assim, a *blockchain* é uma tecnologia de enorme potencial para auxiliar neste roteiro de colaboração interorganizacional, principalmente no contexto dos fluxos de negócios financeiros, em que a disponibilidade exigida e a necessidade de canais digitais *online* são imensas, além da alta exigência de conformidade e auditoria.

Justamente por isso a Fenaban, Federação Brasileira de Bancos, investe em grupos de trabalho para utilização e realização de provas de conceito nessa tecnologia com a intenção dos bancos operarem em regime colaborativo, com rastreabilidade, preservação da privacidade dos clientes e garantia da imutabilidade dos dados compartilhados.

Outro contexto concorrido atualmente em relação às características das soluções bancárias é o das regulamentações de proteção de dados, visto que as entidades financeiras precisam garantir conformidade de suas operações com a Lei Geral de Proteção de Dados (LGPD), regulamentada pela Lei nº 13.709/2018, e assegurar os direitos de privacidade dos clientes.

Esta lei estabelece como coletar, armazenar e compartilhar os dados pessoais e sensíveis de clientes, nos meios físicos e digitais, algo que gera mais complexidade aos negócios bancários, principalmente quando o contexto é o de colaboração interorganizacional. Assim, um sistema distribuído para atender estas necessidades deve manter mecanismos que garantam consenso, segurança e privacidade. Consequentemente, o *blockchain* se mantem como um opção interessante frente essas condicionantes.

2.4 MODELOS DE GOVERNANÇA EM *BLOCKCHAIN*

Fundamental ao contexto de uso do *blockchain* é a definição de quem é o mantenedor da rede. Assim, com relação aos modelos de governança uma *blockchain* pode ser classificada como pública, privada ou híbrida, considerando a característica de permissão que os nós participantes da rede possuem. Na *blockchain* pública todos os nós têm permissão para ler o conteúdo criptografado da cadeia e participar do processo de validação para persistência dos dados. Já na *blockchain* privada apenas um número limitado de participantes pode ler o livro-razão e o processo de validação é centralizado e realizado pela entidade que mantém a rede. E na *blockchain* híbrida todos os nós podem ler a cadeia e a validação para persistência dos dados é compartilhada entre os participantes previamente selecionados, que mantêm aquela rede, justamente por isso esta *blockchain* também é conhecida como de consórcio ou de comunidade.

A questão da governança na *blockchain* está relacionada ao controle de acesso, ou seja, ao tipo de permissão para acesso e registro dos dados [12]. Sendo a decisão sobre a governança da *blockchain* um fator fundamental na definição de sua arquitetura de solução, pois determinará os mecanismos que irão compor esta estrutura e como ela será organizada. Por exemplo, no caso das *blockchains* públicas o investimento nos mecanismos de consenso é mais preponderante do que nas *blockchains* privada, justamente pela questão do poder de decisão dos nós ser muito mais sensível em um contexto de rede aberta.

Outro fator relevante com relação à governança da *blockchain* é o controle sobre o mecanismo de consenso da rede. Assim, na *blockchain* pública todos os nós são equânimes com relação à decisão de consenso na rede, sendo diferenciados apenas pela aplicação do próprio mecanismo, que, por exemplo, pode dar preferência aos nós com mais poder computacional, mas não há uma diferenciação preliminar que beneficie algum nó.

Já na *blockchain* privada a gestão do processo de consenso é concedida por uma organização central, que é a entidade proprietária da rede. E na *blockchain* de consórcio temos um cenário intermediário em que alguns nós são credenciados para realização do mecanismo de consenso pelas entidades que mantém a

blockchain, neste contexto a rede não é aberta e também não é mantida por uma organização central, mas há uma parceria entre entidades para manutenção da *blockchain* e em acordo elas definem a permissão dos nós.

Com o amadurecimento da tecnologia, o avanço em relação ao modelo de referência do *Bitcoin*, o entendimento do *blockchain* como solução tecnológica e a ampliação dos casos de usos corporativos, ficou evidente que a pesquisa e o investimento passaram a ser direcionados às *blockchains* privada e de consórcio, devido à possibilidade de implementação de mecanismos de consenso menos complexos e a aplicação em contextos patrocinados por corporações para uso da *blockchain* como solução tecnológica aos problemas de negócio. Sendo justamente este o cenário considerado para a arquitetura de solução proposta nesta dissertação, que, por isso, adota a especificação de um modelo de consórcio.

2.5 CONSIDERAÇÕES E DISCUSSÕES SOBRE O *BITCOIN*

A maioria das pesquisas sobre *blockchain* ainda estão relacionadas ao *Bitcoin* [16], justamente por esse ter sido o modelo de referência para a tecnologia a partir da sua implementação em 2009, assim é comum que um estudo sobre *blockchain* considere este contraponto. Dessa forma, exponho nesta seção alguns pontos relevantes de discussão sobre o *Bitcoin* antes de aprofundar nas características dos mecanismos da *blockchain*, inclusive, porque o modelo de implementação do *Bitcoin* foi utilizado como referência na comparação com o modelo proposto, conforme cenário explorado no item 4.3.2.

O *Bitcoin* foi proposto no artigo *Bitcoin: A Peer-to-Peer Electronic Cash System* [1] em 2008 por Satoshi Nakamoto [17]. Um trabalho apresentado durante a crise financeira causada pela derrocada do sistema de crédito imobiliário dos Estados Unidos e motivado pela insegurança com relação às instituições financeiras, justamente as entidades autorizadoras das transações financeiras entre duas partes.

Assim, o artigo de Nakamoto propõe um sistema *peer-to-peer* para transferência de valores digitais entre duas partes sem a necessidade de uma entidade terceira autorizar a transação, mantendo os registros dessas em séries de blocos de dados encadeados criptograficamente [1]. Posteriormente, o *Bitcoin* passou a ser explorado para estudos sobre arquitetura de consenso distribuído e se estabeleceu como modelo de referência sobre *blockchain*.

Apesar de não estabelecer o termo *blockchain*, o artigo de Satoshi é o primeiro a descrever a tecnologia posteriormente conhecido por este nome [17]. Ou seja, o artigo estabelece a arquitetura que em alguns anos passa a ser conhecida como *blockchain*, sendo sua implementação também por alguns anos o único caso de referência. Cenário que foi mudando devido à repercussão do estudo e à relevância da criptomoeda *Bitcoin*, que atingiu enorme sucesso principalmente pela disparada do seu valor monetário atrelado ao investimento financeiro como aplicação.

A *blockchain* do *Bitcoin* é pública, pois pode ser acessada por qualquer pessoa que baixe seu livro-razão, e foi estabelecida com a intenção de obter confiança nas interações dos participantes de uma rede distribuída pelo consenso da maioria dos nós e pela transparência do processo. Assim, ao invés de ter uma autoridade central que mantém um banco de dados e protege sua autenticidade, cópias deste ficam distribuídas nos participantes da rede e a persistência dos blocos na cadeia criptográfica ocorre após um

processo de verificação oneroso das transações por prova de trabalho [1].

A segurança da *blockchain* do *Bitcoin* depende do gasto de poder de processamento na verificação de transações. Assim, ele é um sistema mais seguro quanto mais aberto e quanto mais nós da rede gastam poder de processamento na verificação. Este processo no *Bitcoin* é denominado mineração e é realizado pelos mineradores, ou seja, os nós que se dispõem a este trabalho de processamento. Sendo que o processo de mineração do *Bitcoin* garante a validade das transações do bloco e recompensa o minerador por resolver os cálculos criptográficos pela cobrança de taxas das transações, desta forma, a diferença entre o valor de entrada de uma transação e seu valor de saída é a recompensa cobrada pelo minerador que gerou o bloco a ser persistido na cadeia.

Qualquer minerador pode adicionar um bloco válido à cadeia simplesmente publicando-o em uma rede de sobreposição para todos os outros. O bloco é considerado válido quando 50% + 1 dos nós da rede verificarem e validarem o *hash* criptográfico deste, justamente por isso o mecanismo para geração do *hash* é oneroso, mas sua verificação é fácil de ser realizada. No caso de vários mineradores criarem blocos com as mesmas transações e os blocos forem validados pelo processo de verificação pelos nós ocorre um *fork* e a cadeia é bifurcada em ramificações. Subsequentemente, outros mineradores podem adicionar novos blocos válidos a qualquer um desses ramos, visto que mineração pode ocorrer na cadeia principal ou em suas *branchs* [5].

Para resolver os *forks* o protocolo orienta em qual cadeia os mineradores devem minerar, sob o critério que a cadeia vencedora é a mais longa, ou seja, a que exigiu mais poder computacional para ser gerada, e o nó que a gerou vira o líder da cadeia. Ramificações e blocos fora da cadeia principal e suas transações são desconsiderados, podendo ser incluídos na cadeia posteriormente desde que obedecendo as regras de mineração. A disseminação de blocos pela rede de sobreposição do *Bitcoin* leva segundos, enquanto o intervalo médio de mineração é de 10 minutos. Portanto, a bifurcação acidental ocorre em média uma vez a cada 60 blocos.

Em resumo, a *blockchain* do *Bitcoin* fornecem um banco de dados distribuído de governança descentralizada e pública, em que as transações são organizadas em blocos e para ter a transação escrita na cadeia de blocos requer um pagamento na forma de uma taxa de transação. Os nós que participam da *blockchain* seguem um protocolo de eleição de líder para decidir qual nó consegue escrever o próximo bloco e coletar as respectivas taxas de transação, considerando o sucesso na realização do processo de prova de trabalho. Nem todos os nós da rede participam da eleição do líder, apenas os mineradores, sendo que no início de cada rodada todos eles começam a trabalhar em um novo problema de cálculo criptográfico derivado do último bloco, e aquele que primeiro resolver o problema consegue escrever o próximo bloco.

A *blockchain* do *Bitcoin* possui as seguintes características: transparência, consenso, disponibilidade, segurança e descentralização. O *Bitcoin* é transparente porque todos os pares da rede enxergam os blocos e as transações gravadas eles. O conteúdo é protegido por chaves privadas de criptografia, mas os dados são apropriados por todos os participantes. Sendo imutável e irrevogável devido ao processo de validação dos blocos de transações, em que é impossível fazer mudanças na cadeia, ou seja, deletar ou alterar transações registradas na cadeia, devido ao encadeamento dos blocos com base em valores obtidos pelas operações de *hash* criptográficos.

O consenso na *blockchain* do *Bitcoin* é obtido pelo mecanismo da prova de trabalho, em que os blocos

de transações são verificados pelos participantes da rede e não podem ser fraudadas sem um custo computacional descomunal, pois 51% da rede deve ser atacada sincronamente. E possui disponibilidade por não ter um único ponto de falha, visto que caso um nó participante da rede falhe, não há queda do serviço, já que os outros continuarão a operar, mantendo a disponibilidade da informação.

A segurança é obtida pela verificação mútua entre os nós da rede *blockchain* do *Bitcoin* dos blocos de transações, que vai se tornando mais segura conforme mais participantes estão na rede, justamente por isso foi estabelecida a recompensa no processo de consenso, incentivando a participação de novos nós. E a descentralização é uma garantia fundamental para que os participantes da rede transacionem entre si de forma autônoma, assim não há autoridade única que aprove as transações ou garanta regras específicas, os participantes da rede precisam chegar a um consenso para aceitar transações.

A *blockchain* do *Bitcoin* apresenta qualidades contundentes como solução tecnológica, porém, ainda assim, os mesmos mecanismos que proporcionam as qualidades identificadas geram desvantagens, entre as quais: falta de escalabilidade, consumo excessivo de tempo, alto custo computacional, excessiva replicação de dados, alta latência e questões com tamanho e largura de banda.

A falta de escalabilidade e a excessiva replicação advêm da necessidade de manter a cadeia em cada nó para participar da rede, pois a capacidade de armazenamento e poder computacional exigidos dos membros da rede acaba sendo muito grande conforme seu crescimento. O consumo excessivo de tempo e processamento advêm do mecanismo de prova de trabalho, que consome bastante tempo e poder de processamento para resolução dos cálculos criptográficos envolvidos no processo de *hashing*.

Já a alta latência ocorre pela natureza da cadeia que exige *serialização* na inserção de registros e pela complexidade no mecanismo de consenso. E as questões com relação ao tamanho e largura de banda estão relacionadas aos limites de tamanho do bloco de até 1 Mb e a latência de um bloco criado a cada 10 minutos, portanto há uma limitação no número de transações [1]. Em média 500 transações por bloco.

Essas situações devem ser consideradas ao definir novos modelos de uso de *blockchain* para aproveitar os benefícios da tecnologia e contornar as desvantagens. Com o avanço das pesquisas sobre *blockchain* várias dessas questões foram endereçadas e muitos dos *cases* e plataformas disponíveis utilizam mecanismos que superam os problemas de escalabilidade e de desempenho da *blockchain* do *Bitcoin*.

Neste contexto, o amadurecimento da tecnologia vem passando justamente pela ampliação das linhas de pesquisa e implementações de *blockchain* além do *Bitcoin*. Assim, a tecnologia está sendo credenciada como uma arquitetura de solução tecnológica para contextos de redes distribuídas, em que os principais problemas, mas não exclusivamente, sejam: confiança, transparência e imutabilidade.

2.6 MECANISMOS DE CONSENSO UTILIZADOS NAS REDES *BLOCKCHAIN*

Os mecanismos de consenso são estruturas fundamentais para as arquiteturas *blockchain*, bem como para outras soluções distribuídas organizadas em redes *peer-to-peer*. Sendo que, quando do surgimento do *Bitcoin* [1], arquiteturas distribuídas e arranjos *peer-to-peer* não eram novidades, pois já existiam fazia bastante tempo. Porém, o *Bitcoin* apresentou um mecanismo para atingir confiança na rede pela execução

de processos de consenso que tornam mais difíceis os ataques à rede, justamente por isso o maior interesse.

O amadurecimento dos conceitos apresentados e suas implementações ajudaram a credenciar a *blockchain* como solução tecnológica, em grande parte pela confiança conseguida com a utilização dos algoritmos de consenso em um contexto distribuído e os mecanismos de criptografia utilizados para manter a estrutura de dados responsável pelo encadeamento dos blocos.

Os mecanismos de consenso funcionam como métodos para tomada de decisão conjunta na rede *peer-to-peer* do *blockchain*, assim, nesse contexto distribuído, os nós apoiam uma decisão que será exercida para todos, concordando que isso beneficiará a rede. Sendo que existem dois grupos principais de mecanismos de consenso [18]: a) os algoritmos de consenso baseado em provas, do inglês *proof-based consensus algorithms*, que exigem que os nós disputem sobre determinado aspecto para definição dos mais qualificados para execução do trabalho, e b) os algoritmos de consenso baseado em votação, do inglês *voting-based consensus algorithms*, que exigem que os nós da rede troquem resultados da verificação de um novo bloco ou de uma transação para que a definição da maioria determine a decisão final.

Os mecanismos e suas orientações e perspectivas de uso dependem do tipo de algoritmo definido para consenso. Entre os principais e mais importantes para esta trabalho destacam-se os seguintes: prova de trabalho, prova de risco, prova de autoridade e os de tolerância às falhas bizantinas práticas; conhecidos em inglês respectivamente como: *proof-of-work* (PoW), *proof-of-stake* (PoS), *proof-of-authority* (PoA) e *practical byzantine fault tolerance* (PBFT).

2.6.1 Prova de Trabalho

Prova de trabalho é o algoritmo de consenso mais conhecido e utilizado, justamente por ter sido o estabelecido para a *blockchain* do *Bitcoin* [1]. Nele a responsabilidade do consenso recai sobre os nós da rede que tentam resolver problemas matemáticos complexos que exijam muito poder computacional. Os nós que resolverem os problemas solicitam validação pela rede das estruturas de dados geradas com base nas transações delimitadas para o bloco, que, caso validadas, geram ao nó o direito de persistir o bloco na cadeia, sendo recompensado por isto. Ou seja, o nó vencedor constituirá a cadeia mais longa e determinará a atualização da rede caso sua contribuição seja aprovada por esta.

Os algoritmos que utilizam prova de trabalho são do tipo de consenso baseado em provas, os quais no caso são as comprovações das resoluções de desafios matemáticos complexos. Os mecanismos dessa categoria variam com relação ao tipo de problema que estabelecerá a prova do trabalho e no tipo de definição com relação aos nós que podem registrar blocos na cadeia. A recompensa sobre o esforço imputado no mecanismo não é obrigatório, mas é frequente nesse tipo de algoritmo para aumentar o interesse dos nós em gastar poder computacional para resolução dos problemas que constituem a prova de trabalho.

Quanto ao aspecto da definição dos nós que podem gerar blocos na cadeia, a maioria dos algoritmos utiliza a verificação positiva por mais de 50% dos nós ativos da rede para determinar a eleição do nó que registrará o bloco na cadeia, porém alguns algoritmos estabelecem processos diferentes, que consideram parâmetros como sorte e poder na rede para determinar os nós com permissão de verificação no processo de consenso [18]. E quanto a questão dos tipos de problemas estes podem ser relacionados ao cálculo de funções *hash*, a fatoração de números inteiros, a resolução de equações matemáticas complexas, a

determinar as entradas por engenharia reversa a partir de uma saída, entre outros.

Entre as principais vantagens desse tipo de algoritmo está a independência de entidades centralizadas para validação das transações, pois a verificação é realizada pelos nós da rede de maneira equânime. Outra vantagem é proteção da rede, visto que para adulterar a *blockchain* um nó precisa controlar mais de 50% do poder de processamento da rede para garantir que ele seja o primeiro a gerar o novo bloco e estabeleça a cadeia mais longa [19].

Por outro lado, o mecanismo apresenta como desvantagens o alto consumo de energia e os expressivos custos com *hardwares* especializados para execução dos complicados algoritmos matemáticos envolvidos na prova de trabalho. Além da latência de confirmação das transações, pois este complicado processo gasta muito tempo na resolução dos problemas matemáticos, que inclusive não têm seus resultados aplicados em nenhum outro contexto, que não seja como processo do próprio mecanismo.

Outra desvantagem é a concentração dos *pools* de mineração, que são entidades especializadas em solucionar os problemas de prova de trabalho na *blockchain*, os quais concentram a capacidade de geração dos blocos nas redes públicas, como o *Bitcoin*, visto o maior poder de resolução de provas de trabalho em comparação aos nós independentes. Assim, este fator ainda gera uma fragilidade conhecida como o ataque de 51%, em que a rede pode ser comprometida pelo controle dos usuários majoritários, que assumiriam a maior parte do poder de resolução da prova de trabalho, consequentemente, controlando o consenso na rede, podendo monopolizar a geração de novos blocos e receber recompensas, além do poder de reverter transações.

Os principais exemplos de *blockchain* que utilizam o *proof-of-work* como mecanismo de consenso são o *Bitcoin* [1] e o *Ethereum* [20], justamente as maiores plataformas da tecnologia atualmente. Outras aplicações conhecidas que utilizam este tipo de algoritmo são o *LiteCoin* [21], o *Bitcoin-NG* [22] e o *Primecoin* [23].

2.6.2 Prova de Risco

Assim como o algoritmo de prova de trabalho, o mecanismo de prova de risco é muito utilizado em redes *blockchains* públicas, principalmente nas implementações de criptomoedas. Este também é um mecanismo do tipo dos algoritmos de consenso baseados em provas e nele cada nó pode validar blocos, porém as chances de sucesso aumentam proporcionalmente à quantidade de moedas que eles detém, ou seja, conforme o risco assumido na rede.

O algoritmo de prova de risco geralmente trabalha com o conceito de idade da moeda, que seria seu valor multiplicado pelo tempo de existência dela, ou seja, o período após a sua criação. Assim, quanto mais um nó mantem as moedas na rede, mais poder ele detém e maiores as recompensas pelo processo de validação dos blocos.

Desta forma, o mecanismo trabalha com a validação pelo resultado apresentado à problemas matemáticos complexos, porém invés de tratar isso de forma equânime, a dificuldade de problema é inversamente proporcional a quantidade e a idade das moedas do nó. Assim, há um incentivo para os nós manterem as moedas na rede e cumpre-se o objetivo, de quando o mecanismo foi estabelecido, de ser uma alternativa

ao desperdício de poder computacional da prova de trabalho, mantendo a segurança. A ideia é que os atacantes da rede precisem acumular um grande número de moedas e mantê-las por tempo suficiente para atacar a rede *blockchain*, o que dificulta muito o ataque [19].

O uso desse mecanismo não se restringe apenas aos *blockchains* de criptomoedas, pois o conceito da prova de risco pode ser adaptado ao objeto de consenso da rede, seja valores monetários, bens ou informações. A intenção é que o algoritmo seja capaz de relacionar que quanto mais risco sobre os dados da rede o nó possui, mais poderoso seu voto para obtenção de consenso [18]. Sendo que neste tipo de algoritmo cada bloco é validado antes que a rede adicione outro bloco ao registro do *blockchain*, assim as rodadas de votação são melhores estabelecidas.

A principal vantagem do algoritmo de prova de risco advém justamente do seu objetivo de superar a fragilidade da prova de trabalho com relação ao alto gasto de energia, a necessidade de *hardwares* especializados e a ramificação da cadeia. Consequentemente, o algoritmo de prova de risco apresenta eficiência energética, além de ser mais justo com relação a necessidade de poder computacional dos nós, não havendo concentração do consenso nos nós mais poderosos, sendo que há menos ramificações da cadeia, pois as rodadas de definição de consenso são melhores delimitadas.

Por outro lado, o algoritmo não supera a questão da latência de confirmação das transações, afinal mantém na sua arquitetura a resolução de problemas matemáticos complexos e o tempo de validação em cada rodada causa latência na geração dos blocos. E outra questão é o problema da concentração de poder no *pool* de mineração, similar ao que acontece no algoritmo da prova de trabalho, porém a perspectiva dessa concentração de poder não ocorre pelo volume de poder computacional, mas pelo volume de ativos mantidos pelos nós do *pool* na *blockchain*.

O *proof-of-stake* é utilizado nos *blockchains* de criptomoedas *PPCoin* [24], *BlackCoin* [25] e *NextCoin* [26]. E na plataforma *blockchain Stratis* [27].

2.6.3 Prova de Autoridade

O algoritmo de prova de autoridade atinge consenso pela troca de mensagens invés de procedimentos envolvendo resolução de problemas matemáticos. E, diferentemente do algoritmo de prova de risco, que se baseia no volume de ativos da rede mantido pelo nó, a prova de autoridade se baseia na identidade do nó. Assim, alguns nós têm permissão exclusiva para criar novos blocos, protegendo a *blockchain* visto a necessidade de autenticação para exercício dessa permissão na rede.

O mecanismo de prova de autoridade é interessante para as *blockchains* de consórcio, em que algumas entidades autorizadas podem controlar o conteúdo adicionado ao registro público. Esses nós detêm chaves privadas para assinar os novos blocos, agindo como assinantes confiáveis. Assim, todo bloco lido da cadeia pode ser comparado com a lista de assinantes, garantindo que são confiáveis [12].

Na prova de autoridade entende-se que um nó de autoridade está interessado em manter sua reputação para permanecer como tal. Estando a natureza distribuída da rede mantida na prova de autoridade, porque todos os nós de autoridade devem concordar com o estado global da *blockchain* [28], não sendo unilateral a constituição da cadeia.

Assim, o conjunto de nós com autoridade para participar do consenso são os nós que tem permissão para gerar novos blocos e as transações são validadas numa rodada de troca de mensagens entre si, sendo o retorno positivo caso o nó que gere o bloco seja válido para realizar a ação e seu trabalho tenha sido executado corretamente. Justamente devido à este processo, esse mecanismo é enquadrado como algoritmo de consenso baseado em votação.

O consenso no *proof-of-authority* baseia-se em um esquema de rotação de mineração para geração do bloco, uma abordagem utilizada para distribuir de maneira justa a responsabilidade da criação de blocos entre as autoridades da rede [29].

O algoritmo de prova de autoridade resolve a questão da latência sobre o consenso, pois ao invés de depender da resolução de cálculos matemático, ele utiliza a confiança na autoridade dos nós, tendo assim uma resposta de desempenho muito mais rápida que mecanismos como a prova de trabalho e a prova de risco. Neste contexto, outra vantagem da prova de autoridade é não ter alto gasto energético e a necessidade de *hardwares* com alto poder de processamento.

Porém, o mecanismo de prova de autoridade por depender dos nós de autoridade pode ter sua disponibilidade prejudicada caso a quantidade destes não seja suficiente para a rede *peer-to-peer*, e no caso contrário, em que a quantidade de nós seja maior que a necessária, pode incorrer perda da consistência pelo aumento na possibilidade de ramificações na rede [29].

Outra questão é a necessidade inerente de mecanismos que garantam a possibilidade de confiança dos nós de autoridade, que devem ser monitorados para garantir que não possuam intenções maliciosas na rede. Além da impossibilidade de utilização deste tipo de arquitetura em *blockchains* públicas abertas, devido a incoerência de eleição de nós de autoridade em um contexto que todos os nós devem ser iguais.

Apesar de não ser um dos mecanismos mais utilizados, o algoritmo de prova de autoridade é importante para o modelo proposto neste trabalho. Este tipo de implementação foi estabelecida no protocolo *Clique* [30] do ecossistema *Ethereum* para redes privadas [29]. Outro *case* que utiliza o algoritmo de *proof-of-authority* é o *ARCHANGEL*, uma *blockchain* para arquivamento digital de vídeos [31].

2.6.4 Tolerância às Falhas Bizantinas Práticas

O mecanismo *Practical Byzantine Fault Tolerance (PBFT)* é um dos algoritmos do protocolo de tolerância às falhas bizantinas, do inglês *Byzantine Fault Tolerance (BFT)* e têm como ferramenta para consenso a troca de mensagens entre os nós participantes de uma rede *peer-to-peer*. Assim, há alguma semelhança com o mecanismo de prova de autoridade, porém este parte do pressuposto da confiança nos nós, enquanto os protocolos de tolerância às falhas bizantinas estabelecem que não se deve confiar nos nós de uma rede distribuída.

Ou seja, os algoritmos do tipo BFT assumem a possibilidade de falhas na rede e que alguns nós independentes podem falhar em determinados momentos ou atuarem para prejudicar a rede. Desta forma, o algoritmo PBFT também é um mecanismo de consenso do tipo baseado em votação, porém estabelece um processo mais complexo que traz mais segurança à rede, devido a premissa inerente de desconfiança dos nós.

Os protocolos de tolerância às falhas bizantinas devem atingir os seguintes requisitos mínimos: a) cada processo deve ser iniciado em um estado neutro com relação à verificação de consenso, b) deve ser garantido um meio de comunicação entre os nós da rede, c) os nós executam processos determinísticos de verificação para chegar a um de dois resultados, sendo geralmente sim ou não, e d) após um processo de votação por troca de mensagens, os nós consolidam o resultado conforme decisão da maioria.

A tolerância às falhas bizantinas práticas (PBFT) é um dos algoritmos BFT mais bem estabelecidos, que confia em três rodadas de troca de mensagens antes de chegar a um acordo. Garantindo que os nós *3N + 1* possam obter consenso também na presença de *N* nós bizantinos, cuja intenção seja prejudicar a rede [29]. O algoritmo é projetado para sistemas de consenso assíncrono.

Entre as vantagens do mecanismo de tolerância às falhas bizantinas práticas a questão do desempenho em comparação a mecanismos de consenso onerosos utilizados na prova de trabalho e na prova de risco é um fator positivo, visto que os nós se comunicam uns com os outros ao mesmo tempo e chegam a um entendimento do bloco específico sem haver disputa, apenas troca de mensagens. Outra vantagem é a redução de energia e a eliminação da necessidade de *hardwares* com alto poder de processamento, pois não há problemas matemático complexos a serem resolvidos na arquitetura do mecanismo.

Por outro lado, temos como desvantagens a necessidade de manter como participantes do consenso um número reduzido de nós, pois como o mecanismo é baseado na troca de comunicações entre os nós, se houverem muitos participantes eleitos para este trabalho na rede a latência de resposta aumentará proporcionalmente e a disponibilidade da rede corre o risco de ser prejudicada [29]. Consequentemente, pela necessidade de manter um número reduzido de nós para consenso, a rede fica mais suscetível a ataques de manipulação de grupos de nós coordenadamente, comprometendo toda a rede.

O algoritmo PBFT é utilizado em um das plataformas mais disseminadas atualmente, que é o *Hyperledger* [32], além de ser um mecanismo importante para o modelo proposto nesta dissertação. Sendo também muito bem explorado nos trabalho [29] e [33].

2.6.5 Híbridos

Os mecanismos híbridos de consenso são implementados quando arquiteturas de soluções de dois ou mais mecanismos diferentes de consenso são utilizadas para estabelecer um algoritmo único, com o objetivo de manter as vantagens do mecanismo de referência principal e superar suas fragilidades apropriando-se de mecanismos de abordagens diferentes.

Os mecanismos híbridos mais conhecidos utilizam prova de trabalho e prova de risco para estabelecer um algoritmo que utilize o poder de processamento como fator determinante para o consenso, mas que não trate todos os nós como equivalentes, considerando o peso do volume de ativos de cada nó na rede [18], sendo exemplo desse tipo de implementação a criptomoeda *TwinsCoin* [34]. Porém, não há obrigatoriedade da junção de mecanismos ser das referências de prova de trabalho e de prova de risco, pois qualquer junção de mecanismos entre duas ou mais referência estabelece um algoritmo de consenso híbrido.

O objetivo ao se estabelecer um mecanismo híbrido é que os pontos positivos e as vantagens do modelo sejam enaltecidas, assim como, os pontos negativos e as fragilidades sejam superadas. Para isso a gover-

nança da rede, as entidades participantes e os ativos da *blockchain* devem ser considerados na definição dos mecanismos de referência que serão utilizados para implementação deste.

2.7 ESTRUTURA DE ENCADEAMENTO DOS BLOCOS DA *BLOCKCHAIN*

Os registros de um *blockchain* são organizados em blocos vinculados em cadeia, sendo que este vínculo é estabelecido por informações que remetam ao bloco anterior. No mínimo, cada bloco contém o *hash* criptográfico do cabeçalho do bloco anterior, o *timestamp* atual e o *hash* das transições registradas. Assim, o principal vínculo entre os blocos da cadeia que garante a imutabilidade do *ledger* é a estrutura de dados do cabeçalho do bloco.

Os dados armazenados no cabeçalho do bloco na *blockchain* têm as funções de manter os *hashes* do bloco de transação para fins de validação e conter informações adicionais, que variam conforme as diferentes camadas de aplicação ou plataformas de tecnologia *blockchain* [12]. Assim, a estrutura de dados dos cabeçalhos dos blocos descrevem os recursos do sistema para armazenar informações de processamento das transações, consequentemente, da persistência dos registros no *ledger*.

2.7.1 Utilização do *Merkle Tree* em *Blockchain*

A estrutura de árvore de *Merkle*, do inglês *Merkle Tree*, foi utilizada no *Bitcoin* [1] e de longe é a mais conhecida é utilizada em *blockchain* para manter a relação das transações do bloco na estrutura de dados de seu cabeçalho. O mecanismo do *Merkle Tree* tem como propriedades a facilidade de validação de integridade de dados e o auxílio na verificação de inclusão de transações devido à geração de um valor *hash* único que representa todas as transações de dados, assim, se qualquer dado da estrutura mudar, este *hash* único mudará. Ou seja, caso o valor de uma transação seja modificado ou uma nova seja incluída, o valor do *hash* único mudará, o que indicará falha de integridade dos dados.

No *Bitcoin*, por exemplo, a estrutura de dados do cabeçalho do bloco contém as seguintes informações: o *hash* do cabeçalho anterior, o *timestamp* de geração do bloco, o valor de dificuldade de mineração, o *nonce* da prova de trabalho e o *hash* da raiz da árvore de *Merkle* das transações do bloco. Assim, o *hash* identificador de um bloco é aquele do cabeçalho do bloco, que envolve os campos: *Version, Hash-PrevBlock, Hash-Merkle Root, Time, Bits* e *Nonce*, conforme a estrutura representada na figura 2.1. Sendo que, por convenção, o bloco da cadeia mais longa desde o primeiro bloco gerado, conhecido como bloco *genesis*, é considerado o último bloco da *blockchain*, o qual deve ser referência para os encadeamentos de blocos futuros [12].

O campo *Hash-Merlke Root* é um registo de 32 bytes que armazena a *hash* da raiz da árvore de *Merkle*, a qual guarda o cálculo *hash* do somatório de todas as *hashes* das transações do bloco. O campo *Bits* representa a dificuldade do bloco e o *Nonce* é um valor incremental que permite aos nós da rede tornar um novo bloco candidato para que ele seja adicionado à cadeia, executando o algoritmo da prova de trabalho. A ligação da cadeia é estabelecida pelo campo *Hash-PrevBlock*, que mantém a *hash* do bloco anterior e que permite estabelecer ligação com os blocos adjacentes.

Campo	Descrição	Tamanho
Version	Número de versão do bloco.	4
Hash-PrevBlock	*Hash* do cabeçalho do bloco anterior.	32
Hash-Merkle Root	*Hash* raiz de todas as transações do bloco.	32
Time	*Timestamp* corrente.	4
Bits	Dificuldade do bloco.	4
Nonce	Valor incremental para variação do *hash*.	4

Figura 2.1: Estrutura do cabeçalho do *Bitcoin*

O protocolo *Bitcoin* arbitrariamente foi planejado para obedecer à latência de 10 minutos na geração de cada bloco, ao ajustar a dificuldade da prova de trabalho a cada 2016 blocos sob o parâmetro desse cálculo ser atualizado a cada duas semanas, e dos blocos terem tamanhos em torno de 1 Mb. Porém, a característica aberta da rede dificulta esta realização prática, assim como o interesse financeiro dos nós com as recompensas pela geração dos blocos, que motivam a utilização de diferentes abordagens com relação a apropriação de transações nos blocos, considerando seus valores monetários ou tamanhos em *bytes*.

Os *hashes* do *Bitcoin* são gerados com o algoritmo SHA256, que processa mensagens com 264 *bits* e trabalha com palavras de 32 *bits*. A entrada da função de compressão possui 512 *bits* e sua variável de estado contém 256 *bits*, que, por sua vez, gera outra variável com 256 *bits*. O problema criptográfico específico é um *hash* do cabeçalho do bloco cujo resultado deve ser menor que o valor do *hash* anterior e é resolvido pelo processo exposto na geração do cabeçalho do bloco, conforme o mecanismo consegue incrementar com mais velocidade o *Nonce* para variar o *hash* do bloco para resolver esta prova de trabalho.

As *Merkles Tree* representam estruturas de dados como uma árvore de informações resumidas sobre um pedaço de dados maior. Seu principal uso é garantir que blocos de dados recebidos de outros pares em uma rede *peer-to-peer* sejam recebidos intactos e inalterados e verificar se os outros pares não enviaram blocos falsos. Todas as transações na *blockchain* são identificadas através de um *hash*. A raiz da árvore é formada pela *hash* da *hash* de todas as transações organizadas em forma de pirâmide. Apenas a raiz da árvore de *Merkle*, do inglês *Merkle Root*, é incluída no cabeçalho do bloco, o que permite verificar a integridade de todas as transações. Qualquer alteração ou tentativa de alterar uma transação anterior será invalidada e rejeitada, garantindo a imutabilidade na rede.

No *Bitcoin* a *Merkle Tree* é conseguida pela implementação de uma função recursiva, em que um vetor armazena as transações para ser iterado por um par de índices, já que cada ramo terá duas folhas associadas. A função então retorna o *hash* do somatório de dois *hashes*, que é adicionado à árvore *Merkle*, sendo a operação repetida até se chegar ao *Merkle Root*. Com o *hash* do *Merkle Root* e as demais informações que formam o cabeçalho do bloco, o mesmo é submetido para validação pela rede. Para ser considerado um bloco válido a maioria da rede deve conferir e validar o *hash* do bloco. A verificação é realizada pelo *hash* do *Merkle Tree* ao conferir que a transação corresponde aos valores dos *hashes* dos ramos aos quais ela esteja ligada em cada nível da árvore de *Merkle*, conforme a figura 2.2 que mostra a associação da transação Tx03 ao *Merkle Root* como passível de verificação pela informação deste, já que há uma

dependência associativa dela à raiz [1].

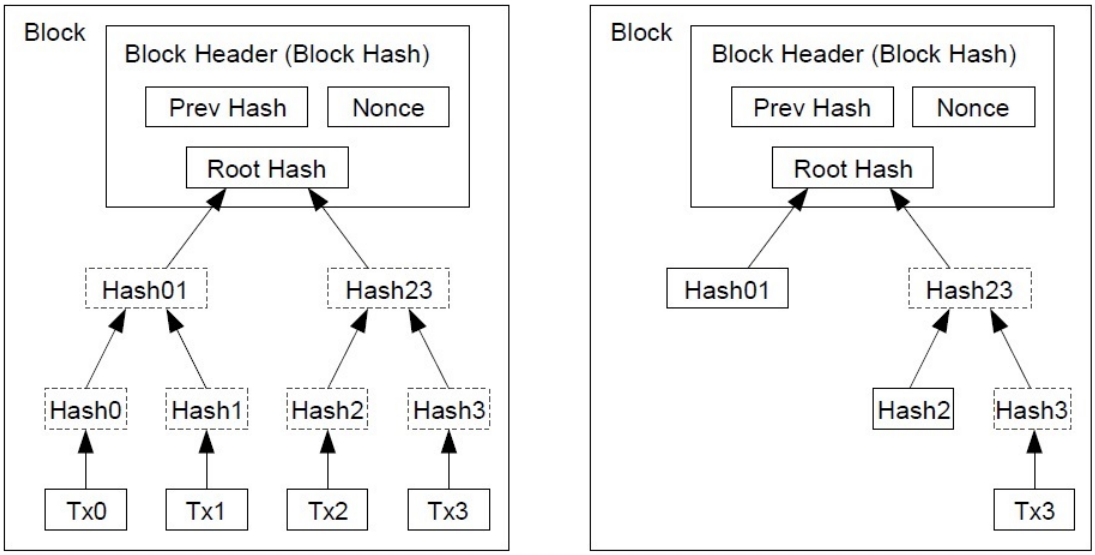

Figura 2.2: Associação da transação Tx03 ao *Merkle Root* [1]

A estrutura árvore de *Merkle* demanda um esforço muito grande para a geração do *hash* raiz, já que este envolve várias operações de *hash*. Por exemplo, em um bloco de apenas 4 transações são necessárias 7 operações de *hash* para se obter a *Merkle Root*, conforme representado na estrutura 2.3. Em um bloco com cerca de 500 transações, a média de um bloco do *Bitcoin*, são executadas mais de 1000 operações de *hash* para chegar ao *Merkle Root*.

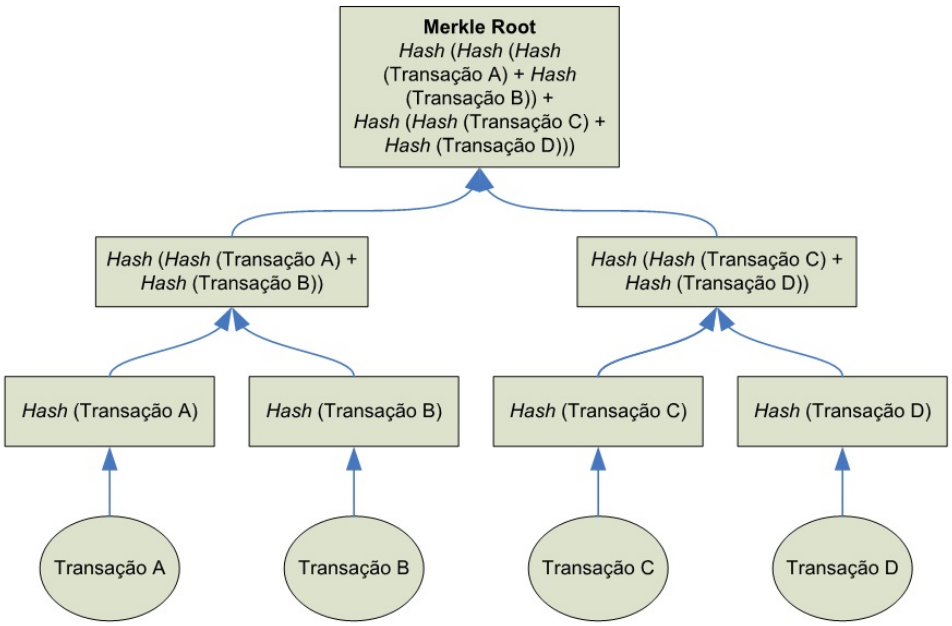

Figura 2.3: Estrutura *hash Merkle*

2.7.2 Considerações sobre o *Patricia Merkle Tree*

A estrutura *Patricia Merkle Tree*, como é conhecido o mecanismo do Algoritmo Prático para Recuperar Informações Codificadas em Alfanumérico, do inglês *Practical Algorithm To Retrieve Information Coded In Alphanumeric*, é uma ferramenta para validação de uma estrutura de árvore de *hashes* sem a necessidade de executar todas as operações para determinar a raiz da *Merkle Tree*, tornando a validação de transações mais rápida e flexível do que o modelo da árvore de *Merkle* tradicional. Sendo a plataforma *Ethereum* [20] o principal exemplo de implementação *blockchain* que utiliza o mecanismo *Patricia Merkle Tree*.

A intenção com o algoritmo *Patricia Merkle Tree* é estabelecer um meio flexível de armazenamento, indexação e recuperação de informações em arquivos grandes, economizando espaço de indexação e tempo de reindexação, mantendo o índice da árvore de transações atualizado mesmo após operações de inserção, atualização, edição ou exclusão, sem precisar recalcular a árvore inteira [35]. Neste contexto, duas premissas são consideradas: a de que profundidade da árvore é limitada e a de que o valor da raiz da árvore depende apenas dos dados, não da ordem em que as atualizações são feitas.

A *Patricia Merkle Tree* é um algoritmo que constrói um índice para uma biblioteca codificada binária de um conjunto de dados, utilizando-o para recuperar informações buscadas por um usuário para as quais as chaves do índice estão endereçadas e mantendo o índice atualizado para refletir as alterações na biblioteca. O mecanismo é eficiente e atinge esses objetivos com quantidades de memória e computação quase mínimas para o problema definido.

A informação buscada pelo usuário pode ser uma transação ou um dado de um contexto maior, independentemente, esta é a chave buscada. Idealmente, conforme a biblioteca cresça, a quantidade de esforço necessário para encontrar as ocorrências de uma chave específica ou para atualizar o índice devem permanecer limitados. O limite deve depender apenas da chave de busca e o número de ocorrências na biblioteca, independentemente do tamanho da biblioteca. A codificação do mecanismo utilizando um alfabeto binário permite que técnicas sejam utilizadas para atender este requisito, visto que o índice definido no *Patricia Merkle Tree* não inclui chaves ou texto, apenas *hashes* que designam locais no texto ou no índice [35].

Uma qualidade do *Patricia Merkle Tree* é sua capacidade de atualizar o crescimento da biblioteca de forma que esta não seja linear ao tamanho do índice acumulado anteriormente, ou seja, o índice é atualizado mantendo sua profundidade, não há uma aumento proporcional deste conforme cresce a biblioteca. Diferentemente do *Merkle Tree*, em que quanto maior a biblioteca, mais difícil gerar o *hash* único dos dados. Inclusive, é importante ressaltar que essa tecnologia permite que blocos fora da cadeia mais longa contribuíam ao processo de validação, construindo uma confirmação de sistema menos centralizada [12].

Assim, os principais motivos para o uso do *Patricia Merkle Tree* apesar de sua complexidade em comparação ao *Merkle Tree* são: a) a velocidade de recalcular o *hash* único após operações de inserção, edição, atualização ou deleção, b) a limitação de profundidade, prevenindo ataques de *DDoS* pela ampliação da biblioteca numa tentativa de tornar as atualizações mais lentas, e c) dependência apenas dos dados para calcular o *hash* único e não da ordem das operações de atualização.

O mecanismo *Patricia Merkle Tree* utilizado no *Ethereum* [20] fornece uma estrutura de dados persistente para mapear os dados binários em um comprimento arbitrário, definindo os termos para manter este índice atualizável de 256 *bits*. O objetivo do protocolo especificado é fornecer um valor único que

identifique um determinado conjunto de pares de valores-chave, que pode ser uma sequência de 32 *bytes* ou a sequência de *bytes* vazia. Esta implementação permite armazenar e manter a estrutura da árvore de uma maneira que se atinja a realização efetiva e eficiente do protocolo [20].

A árvore no *Ethereum* é uma estrutura de dados ordenada que armazena os valores-chave mapeados, em que a chave é o caminho pela árvore para obter o valor correspondente, permitindo que chaves que começam com a mesma sequência tenha valores próximos na árvore, com a desvantagem apenas nos casos em que nenhuma chave compartilha o prefixo. Os nós são referenciados pelo *hash*, sendo quatro os tipos possíveis: a) nós vazios, que são os nós em branco, b) nós folhas, os quais são os nós padrões com uma chave e um valor, c) nós ramos, que são listas de tamanho 17, cujos os primeiros 16 elementos correspondem os caracteres hexadecimais que podem haver na chave e o último representa se há um par valor-chave onde ela termina no ramo, e d) nós de extensão, os quais são os valores-chave cujo valor é o *hash* de outro nó [35]. Conforme representado na estrutura 2.4.

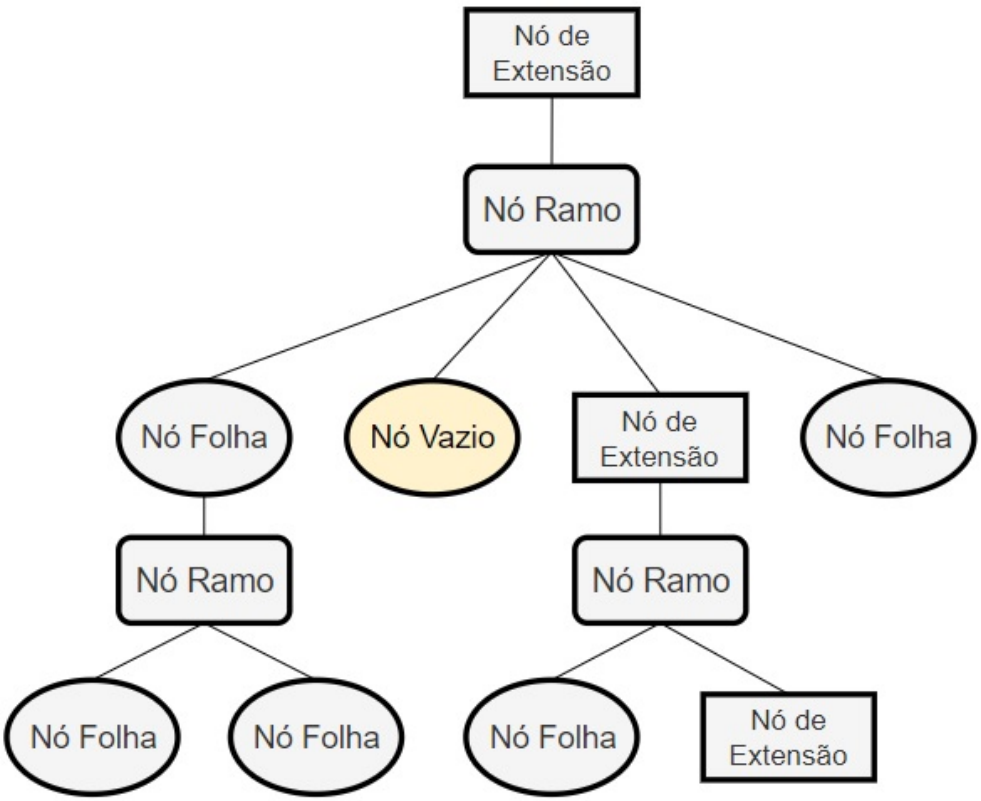

Figura 2.4: Estrutura *hash Patricia Merkle Tree*

Na atualização da árvore as seguintes regras devem ser obedecidas: a) se ao percorrer a árvore parar em nó vazio, deve-se adicionar um nó folha com o caminho que falta e substituir o nó vazio com o *hash* do novo nó folha, b) se ao percorrer a árvore parar em um nó folha, deve-se converte-lo em um nó de extensão e adicionar um novo nó ramo e um novo nó folha, e c) se ao percorrer a árvore parar em um nó de extensão, deve-se converter em um outro nó de extensão com um caminho mais curto e criar novos nó ramo e nós folhas. A estrutura de armazenamento de *hash* da *Patricia Merkle Tree* é representada na estrutura 2.5 pelo exemplo de armazenamento das palavras latinas: *romane*, *romanus*, *romulus*, *rubens*, *ruber*, *rubicon* e *rubicundus*.

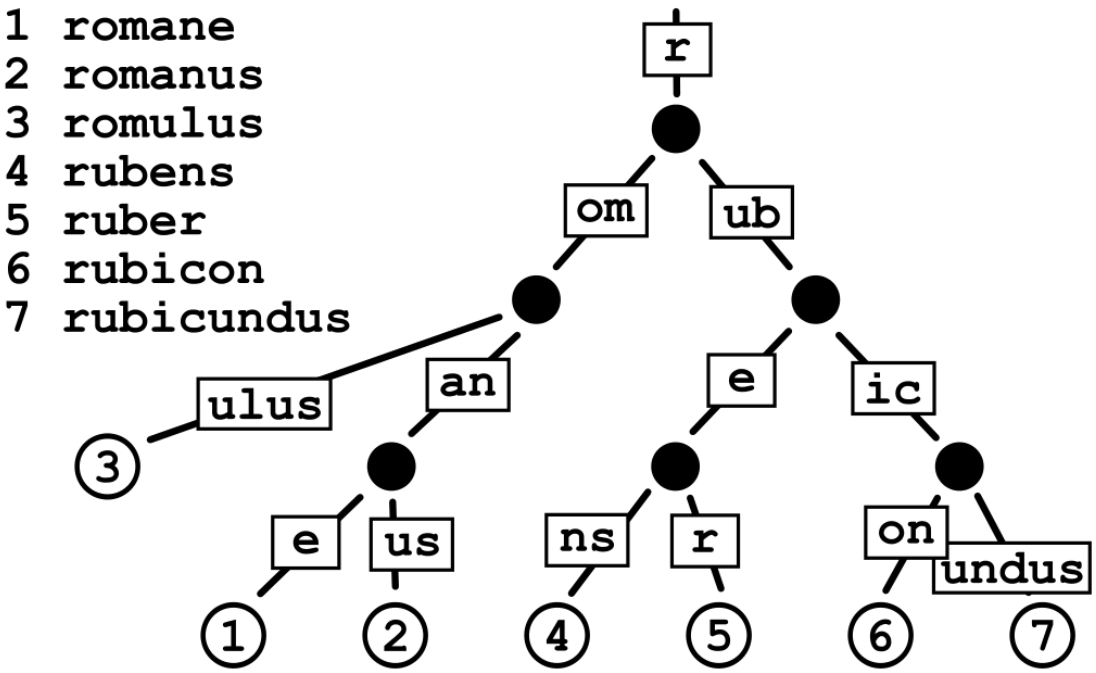

Figura 2.5: Exemplo de estrutura de armazenamento *Patricia Merkle Tree* [2]

2.8 ATRIBUTOS DE SEGURANÇA E PRIVACIDADE EM *BLOCKCHAIN*

As qualidades inerentes com relação à segurança e à privacidade da arquitetura *blockchain* estão vinculadas à sua natureza descentralizada criptográfica, ou seja, à integração entre as tecnologias de: a) armazenamento distribuído, b) organização em rede *peer-to-peer*, c) infraestrutura de chaves públicas (ICP), do inglês *public key infrastructure (PKI)*, e d) mecanismos de consenso [36].

A arquitetura *blockchain* apresenta a vantagem de ser inviolável devido ao seu mecanismo de estrutura de dados, em que as informações das transações são encadeadas criptograficamente considerando o *timestamp* de geração após consenso da rede, assim qualquer modificação de dados anterior ao *timestamp* de referência não é permitida.

Outra vantagem é a recuperação a desastres da arquitetura, que realiza o armazenamento dos dados de forma síncrona nos nós da rede *peer-to-peer*, desta forma, apesar de gerar redundância, a confiabilidade e a tolerância às falhas são melhoradas, pois o ataque em um ou mais nós não causa prejuízo em toda a rede, caso não afete a maioria desta.

Mais uma vantagem do *blockchain* é que ele adota um mecanismo de criptografia assimétrico que permite aos usuários criptografarem dados com sua própria chave privada, sendo que o valor do *hash* da chave pública de um usuário é calculado e executado como um indicador do usuário, não havendo uma relação direta com a identidade do usuário, mantendo a privacidade da informação, pois um adversário não consegue descobrir a chave pública de um usuário a partir do seu endereço público e o cálculo da chave privada a partir da chave pública é muito difícil pela criptografia assimétrica [36].

Assim, a privacidade em *blockchain* é garantida pela identificação do usuário da rede como um identificador *hash* da chave pública sem vinculação com sua identidade no mundo real, em um contexto em que a

característica distribuída do sistema complementa a privacidade, pois não há necessidade de uma entidade centralizada que mantenha as credenciais do usuário e, consequentemente, tenha acesso e controle dos dados deste. Na *blockchain*, os usuários são responsáveis por suas próprias chaves privadas, o que significa que uma chave privada é gerada e tratada pelo usuário em vez de um terceiro, e cada nó servidor armazena apenas fragmentos criptografados de dados dos usuários, sendo as transações até certo ponto anônimas.

2.8.1 Autoridade de Certificação e Modelo Federado

As autoridades de certificação, do inglês *certification authority (CA)*, são entidades, públicas ou privadas, com responsabilidade de emitir, distribuir, renovar, revogar e gerenciar certificados digitais, que é um documento digital que mantém uma chave pública que identifica uma pessoa, física ou jurídica, no mundo real. Conceitualmente, esta intenção está distante dos sistemas *blockchain*, os quais estabelecem mecanismos para independência de terceiros para qualquer tipo de autorização relacionada às ações na rede, além de ter como modelo de referência a utilização de infraestrutura de chave pública mantidas pela própria rede *peer-to-peer*, em um contexto criptográfico assimétrico.

Porém, as tecnologias *blockchain* e de autoridades de certificação podem complementarem-se mutuamente, visto que a certificação permite identificar os indivíduos criptograficamente, registrando com segurança suas manifestações de vontade realizadas com o uso de certificados digitais, um contexto que pode ser incorporado a qualquer sistema *blockchain*. Reciprocamente, as soluções *blockchain* podem ser utilizadas pelas infraestruturas de chave pública para aumentar a disponibilidade e os processos de verificação com utilização de um *ledger* distribuído.

O uso de *certification authorities* em soluções *blockchain* é um mecanismo interessante nas redes com governança privada e de consórcio, visto a possibilidade de utilização deste como uma ferramenta para identificação dos nós na rede, garantindo que as intenções das entidades que mantêm a rede sejam prevalecidas, podendo as ações dos nós depender de verificação de seus certificados digitais, estabelecendo inclusive suas permissões conforme sua participação na rede, funcionando como um importante mecanismo para estruturar camadas de acesso em redes *blockchain* privada e de consórcio.

O modelo federado é um mecanismo para gerenciamento de identidades em sistemas de informação com abordagem descentralizada, distribuindo o papel de autenticação a diferentes domínios de um sistema invés de depender de uma entidade centralizada, que mantém todas as identidades válidas e possui alto risco de vir a ser um ponto de falha. Assim, o modelo federado otimiza a troca de informações relativas a uma identidade de usuário se baseando em uma relação de confiança entre organizações.

Este modelo é recomendado para *blockchain* de consórcio, pois é um padrão na arquitetura corporativa, permitindo a interoperabilidade e o compartilhamento de informações entre linhas de negócios e sistemas de tecnologia da informação. Como a *blockchain* de consórcio é motivada pela organização de parcerias entre as entidades que mantêm a rede, concordando em uma gestão descentralizada com o objetivo de beneficiar todos os participantes, o modelo federado é uma solução que habilita a possibilidade de autenticação descentralizada que permita a um usuário transitar entre domínios da *blockchain* mantidos por diferentes organizações.

2.8.2 Privacidade de Dados Incorporada e Adicional

Alguns sistemas em *blockchain* trabalham com uma política de privacidade de dados inerente à rede, assim neste as informações são ocultadas por padrão, mantendo todas as comunicações trocadas pelos nós privados. Um exemplo desse tipo de implementação é o sistema *Zerocash*, no qual todos os dados publicados nessa *blockchain* pública, como remetente, destinatário e valor de uma transação permanecem privados [37]. Outro exemplo é a plataforma *blockchain Corda*, que tem como característica as transações criptografadas em que apenas as partes envolvidos na operação tenham acesso aos dados [38].

Nestes sistemas as implementações possuem uma preocupação arquitetural de manter a privacidade dos dados, indo além das técnicas de criptografia geralmente utilizadas nas *blockchain*. Sendo inclusive comum no *blockchain* público o artifício do pseudo-anonimato com a utilização de *hashes* para identificação em substituição à contas de usuários, assim os usuários ficam escondidos de outros nós e se mantém anônimos ao sistema [39].

A privacidade é uma preocupação comum das soluções *blockchain* para atender a exigência de que todos os dados pessoais e as informações sensíveis permaneçam confidenciais, ou seja, não públicas, além de que o acesso à elas possa ser controlado pelos seus proprietários, ambas questões sensíveis no contexto de soluções distribuídas.

Assim, a necessidade de privacidade ou confidencialidade é conquistada utilizando técnicas de criptografia, por isso, no modelo mais básico, a transação é encriptada pelo remetente com a chave pública do destinatário e desencriptada por este com sua chave privada. Somente o destinatário ou alguém que tenha sua chave privada será capaz de desencriptar e entender os dados [39].

Porém, os sistemas em *blockchain* podem ir além e utilizar mecanismos em sua definição arquitetural que garantam essa privacidade ou recorrer à soluções externas para ocultar as informações, justamente o caso das *blockchain* que possuem privacidade de dados adicional.

Estes mecanismos adicionais podem ser algoritmos que ofuscam as informações para que elas façam sentido apenas dentro do contexto do sistema e, caso utilizada por entidade maliciosa, não seja tão simples obter seu significado. Sendo um exemplo a *CoinJoin* [40], cujo algoritmo agrupa diversos pagamentos e mistura os remetentes e os destinatários para dificultar saber a origem de cada transações.

Outro exemplo é a implementação de camadas de controle de acesso aos dados, em que os usuários autorizados acessam aos dados permitidos e técnicas de criptografia impedem que usuários sem autorização acessem, sendo um contexto relacionado às *blockchains* privada e de consórcio devido a necessidade de uma autoridade de certificação que mantenha a lista de controle de acesso.

O modelo de privacidade de dados adicional permite que os mecanismos que serão utilizados na arquitetura do *blockchain* para confidencialidade dos dados sejam definidos conforme a necessidade específica do sistema considerando o sigilo da informação armazenada, além de uma análise mais particular da implementação para evitar problemas de disponibilidade, escalabilidade ou desempenho.

Assim, essas implementações consideram a governança do *blockchain*, as características dos nós participantes da rede, o tipo de relacionamento entre os nós e os respectivos graus de confiança, a robustez do mecanismo de consenso adotado, a complexidade das estruturas de dados da cadeia, entre outros aspec-

tos, para determinar mecanismos que garantam a privacidade dos dados sem interferir no funcionamento distribuído da rede.

2.9 REGULAÇÕES BANCÁRIAS E *BLOCKCHAIN*

O cenário bancário é caracterizado por estar submetido a rígidos processos de *compliance* e atuação forte de órgãos reguladores, que supervisionam e centralizam as ações dos atores envolvidos nas operações do sistema financeiro. Contudo, a aplicação de *blockchain* no sistema financeiro carece de regulação e provavelmente forçará repensar o papel dos reguladores e sua forma de atuação.

As diretrizes de regulação bancária foram estabelecidas conforme o avanço do sistema financeiro em um esforço de cooperação coordenado internacionalmente culminando em diversos acordos sobre transparência bancária, sonegação de impostos e troca de informações financeiras e fiscais [41]. Neste sentido, a aplicação de *blockchain* no âmbito bancário deve considerar o esforço de adequação às obrigações regulatórias, porém, em contrapartida, os órgãos reguladores devem compreender que estas não podem travar o desenvolvimento de soluções e o avanço da tecnologia.

O principal obstáculo neste contexto é justamente as características da *blockchain* de resiliência e inviolabilidade, que tornam difícil desativar o sistema ou alterar sua implementação para atender modificações por medidas regulatórias ou decisões judiciais. Esse cenário representa um desafio para a regulação financeira e para o trabalho dessas autoridades em todo o mundo, somado às características de anonimato das *blockchains* públicas, que incentivam a ação de pessoas que estejam interessadas em encobrir propósitos ilícitos [42].

Atualmente, a principal abordagem adotada pelos reguladores tem sido aguardar e acompanhar a evolução do crescimento dos negócios que utilizam a tecnologia *blockchain* em soluções do sistema financeiro, adotando medidas regulatórias específicas apenas após maturidade dos modelos de aplicação. Essa abordagem tem a vantagem de impor obrigações regulatórias exclusivas somente quando os participantes serão grandes e maduros o suficiente para suportar o ônus da regulação imposta, além de eliminar as barreiras regulatórias de entrada que podem inibir o empreendedorismo e a inovação. Mais que uma abordagem passiva, esta é uma estratégia cuidadosa e ponderada de estudar o assunto antes de emitir uma nova legislação ou alterar um marco regulatório. Aproveitando o período para consultar partes interessadas e especialistas, coletar informações do mercado e realizar pesquisas para avaliar os resultados em diferentes nichos de atuação e vários países [43].

Sob uma perspectiva mais ativa, duas outras abordagens também vêm sendo utilizadas, que é a regulamentação por recompensas [44] baseada em proporcionar um benefício aos regulados que cooperem e enveredem esforços para cumprir os requisitos regulatórios existentes, e o *Sandbox* Regulatório [45] que permite aos regulados testarem seus produtos e serviços com um número limitado de usuários sob a supervisão dos reguladores sem ter que cumprir todas as regras existentes para o setor. A primeira abordagem possui a dificuldade de que os reguladores devem saber de antemão quais normas regulatórias eles querem aplicar, um ponto de dor contundente para *blockchain* e novas tecnologias. Já a segunda abordagem não possui essa desvantagem, enquanto apresenta um cenário mais propício para aprendizado e teste, trazendo

segurança jurídica de início, além de um canal de diálogo e cooperação [43].

Como a utilização da tecnologia *blockchain* em soluções do sistema financeiro é uma questão recente ainda há muito o que evoluir com relação ao contexto regulatório, tanto para que as soluções estejam aderentes às regulamentações bancárias nacionais, como as normas e regulamentações do Banco Central - BC e do Conselho Monetário Nacional – CMN, e internacionais, como as recomendações do Comitê de Basileia para Supervisão Bancária (*Basel Committee on Banking Supervision - BCBS*), quanto para utilização dos mecanismos da *blockchain* como ferramentas para implementação e otimização dos controle de *compliance* bancário.

2.10 AUDITORIA NO CONTEXTO DAS OPERAÇÕES BANCÁRIAS

As operações bancárias, tanto por regulamentações, quanto pelas características financeiras, devem ser permanentemente auditadas para verificar o alinhamento com questões legais e de conformidade. Assim, soluções e sistemas bancários mantêm mecanismos para auditoria constante, especialmente em um contexto de interoperações, em que diversos atores participam de um processo de negócio e devem haver processos para garantir que todos obedecem normas de *compliance*, sigam regulamentações internas e externas e evitem prejuízo de qualquer uma das partes.

No sistema financeiro as atividades de auditoria são tidas como essenciais e contribuem para ambientes de negócios caracterizados por maior confiança e credibilidade, que podem ser obtidas com transparência, monitoramento e simetria. Este contexto apresenta-se como uma oportunidade de uso das soluções de *smart contracts* em *blockchain*, visto a transparência inerente aos mecanismos para descentralização e imutabilidade, além de que os contratos inteligentes sendo autoexecutáveis podem garantir o monitoramento constante e, como algoritmo codificado, há simetria em suas decisões.

A organização de forma distribuída do *blockchain* traz transparência, disponibilidade e descentralização para a auditoria. Essas características são fundamentais para os incentivos das entidades financeiras em soluções *blockchain*, principalmente por apresentar um contexto em que há a possibilidade de manter auditorias internas e externas de forma contínua. Desta forma, a *blockchain* proporciona maior eficiência no processo de contabilização e escrituração de transações podendo promover auditoria automática e contabilidade contínua, além de reduzir tempo e custo desses serviços e dar maior transparência, acurácia e velocidade pela manutenção do *ledger* compartilhado, reduzindo fraudes e aumentando a integridade das informações [46].

2.10.1 Considerações sobre *Smart Contracts*

O conceito dos contratos inteligentes, do inglês *smart contracts*, foi apresentado primeiramente por Nick Szabo no artigo *Smart Contracts: Building Blocks for Digital Markets* [47] em 1996, estabelecendo-os como instruções digitais sob a ideia básica de que muitos tipos de cláusulas contratuais podem ser incorporadas ao *hardware* e *software* de maneira a possibilitar diligência automática nos casos de quebra de contrato.

Considerando o artigo, um contrato possui quatro objetivos definidos: a) **monitoramento**, que é a possibilidade de uma instância de observar o desempenho de terceiros e prová-lo a outras instâncias, b) **confiabilidade**, que é a capacidade de uma instância de provar a um árbitro que um contrato foi executado ou violado ou a possibilidade do árbitro descobrir por ele mesmo, c) **privacidade**, que é o princípio de que o conhecimento e o controle sobre o conteúdo e a execução de um contrato deve ser distribuída entre as partes necessárias para a executá-lo, e d) **aplicabilidade**, que é minimizar a necessidade de forçar a aplicação dos acordos, visto que a obrigatoriedade é inerente ao contrato.

Neste cenário a utilização de algoritmos executados em um contexto distribuído cumprem aos objetivos para garantir a coerente execução de contratos de forma automática. A solução *blockchain* como um sistema distribuído *peer-to-peer* dotado de mecanismos de criptografia e consenso foi referenciada como opção por sua contabilidade descentralizada para implementação de contratos inteligentes, que podem ser convertidos em código de computador, armazenados e replicados [12].

Os contratos inteligentes podem ser definidos estritamente como algoritmos de autoverificação, autoexecução e autoimposição de respostas, armazenados e protegidos na *blockchain* [48]. Nos *smart contracts* os relacionamentos entre as partes são registrados como operações de criptografia. Eles são autoexecutáveis, codificados em termos condicionais, sem a necessidade de uma terceira parte interveniente e mantidos por registros imutáveis acessíveis de forma transparente. Além de que contratos inteligentes baseado em *blockchain* permitem implementar organizações autônomas descentralizadas, do inglês *Decentralized Autonomous Organization (DAO)* para colaboração entre empresas [49].

Como contratos inteligentes são programas imutáveis implantados e executados de forma descentralizada na *blockchain*, há várias possibilidades de casos de uso relacionadas a processos de execução automática e independente. Entre as principais aplicações temos: a) utilização para armazenar registros, liberá-los e renová-los automaticamente, b) automatização da execução de um fluxo de eventos de uma cadeia de processos, c) transparência na manutenção e transferência de bens e ativos, d) registro de termos e condições de contratos bilaterais de forma imutável e transparente, e) proteção de diretos de propriedade, e f) execuções de regras de conformidade e monitoramento. Assim, os *smart contracts* são ferramentas poderosas para operacionalização de processos de auditoria, principalmente no contexto das operações bancárias, cujas as regras são transparentes e o volume transacional muito alto.

3 PROPOSTA DE MODELO DE *BLOCKCHAIN* PARA OPERAÇÕES INTERBANCÁRIAS

Diariamente várias entidades financeiras realizam operações entre si, que são comunicadas a uma autoridade central, a qual garante o funcionamento do sistema como um todo. A maioria dessas transações são comandadas de forma *online*, mas sua efetivação só acontece várias horas depois, após o processamento em lote das transações de determinado produto, que geralmente ocorrem fora da janela de atendimento comercial.

O mecanismo de interface mais comum ainda entre instituições bancárias envolve os arquivos sequenciais. Assim, as transações de determinado produto vão sendo realizadas durante uma determinada janela de horário comercial para serem agrupadas em um arquivo de interface a ser transmitido para outra instituição, que fará o processamento de cada transação aplicando as regras de negócio relacionadas e enviará um outro arquivo de retorno com o resultado desse processamento à instituição de origem. Desta forma, o comum é uma transação realizada hoje ser efetivada apenas nos próximos um ou dois dias úteis subsequentes.

Em muitos desses casos a autoridade central de verificação apenas recebe esses movimentos mensalmente e as auditorias de conformidade e análise de fraude são realizadas meses após a transação ter sido realizada. Logo, os mecanismos atuais de operações interbancárias são entraves para auditoria constante, verificação de conformidade permanente, efetivação imediata do serviço contratado, transparência das ações de contratação e economia dos valores gastos com as verificações pela autoridade central.

Assim, este trabalho de dissertação apresenta um modelo alternativo com utilização da tecnologia *blockchain* para comunicação das operações interbancárias entre as entidades parceiras participantes de uma rede. As características da *blockchain* de descentralização, imutabilidade e disponibilidade são fundamentais nesse contexto em que há necessidade de modernizar o processo de comunicação para eliminar inúmeros pontos de falha, diminuir a latência de resposta, reduzir a replicação de processamento e operação e possibilitar auditoria permanente. Bem como promover o compartilhamento de informações de forma *online* e integrada, o que garante transparência e economia.

O modelo proposto possui o objetivo de utilizar a rede *blockchain* para substituir as soluções legadas de comunicação de operações interbancárias, constituindo uma arquitetura cujo os mecanismos garantam melhor escalabilidade, desempenho e privacidade para que haja robustez nesses pontos. A intenção é garantir vantagem sobre o processo legado das operações interbancárias com o uso da tecnologia *blockchain*, possibilitando que qualidades como transparência, autonomia de monitoramento e verificação, disponibilidade, rastreabilidade e integração sejam incorporadas ao contexto das comunicações entre entidades financeiras.

Além das vantagens inerentes a redução de custos pelo compartilhamento da solução e a dispensa da verificação por terceiros, há uma diminuição do trabalho de *backoffice* exigido pelas soluções legadas, que são complementadas por processos de conformidade no acompanhamento das operações interbancárias e seus reflexos, para atender às necessidades negociais e regulatórias.

3.1 OPERAÇÕES INTERBANCÁRIAS

As operações interbancárias são as transações comandadas entre as entidades financeiras para transferência de valores, garantias e contratos, processamento de operações de produtos e soluções financeiras, e comunicações sobre estados de operações e processos executados de forma conjunta entre as entidades parceiras.

Estão no rol das operações interbancárias, por exemplo, processamento de pagamentos, transferência de reservas monetárias, garantias de operações de derivativos, operações de crédito e seguro e portabilidade de contratos de crédito ou serviço, além daquelas específicas de determinados produtos bancários compartilhados, em que o operador é uma entidade e o balcão de venda, outra.

As operações interbancárias possuem características sensíveis por estarem relacionadas a manutenção de ativos e serviços financeiros de clientes, que são os detentores dos bens ou direitos, mas que confiam na segurança da execução desses processos pelas entidades contratadas. Justamente por isso o contexto de regulamentação é extenso e complexo, as questões de privacidade e segurança são pontos fundamentais de atenção, assim como, disponibilidade do sistema de comunicação, persistência das transações, transparência das operações e auditoria dos dados histórico. Sendo comum a fiscalização por órgãos reguladores e o investimento em mecanismos de controles internos.

3.1.1 Problemas dos Modelos Atuais

As soluções tradicionais legadas de comunicação interbancária não conseguem estabelecer uma arquitetura compartilhada como a de *blockchain*, tendo dificuldades para garantir disponibilidade, transparência e auditoria, sendo necessário investir em processos onerosos e externos ao sistema para tentar superar estes problemas.

As soluções *blockchains* para uso como em transações financeiras anônimas, semelhante às criptomoedas, no contexto bancário ainda carecem de regulamentação e autorização, porém a *blockchain* como solução tecnológica para comunicação de operações interbancárias entre instituições financeiras é um possibilidade real e uma ótima opção em substituição ao cenário tradicional.

No modelo legado, por exemplo, quando contratamos um seguro, na maioria dos casos, uma corretora gera um boleto, que tem como beneficiário a própria corretora ou o banco operador. O pagamento do boleto até a data de início da vigência do seguro gera o direito em relação ao acordo estabelecido, mas não é a efetivação da transação de contratação do seguro. Esta ainda dependerá do processamento pelo mantenedor do seguro, do registro na entidade reguladora dos acordos de seguro e dos relativos retornos. E em geral toda a comunicação é feita por interfaces de processamento de arquivos em lotes, assim o seguro contratado terá sua real efetivação dias após a realização do acordo entre as partes, vendedor e comprador.

Neste e em outros acordos financeiros a ação que o usuário final entende como a de efetivação do acordo apenas gera um direito condicional. No exemplo acima, caso uma das interfaces apresente algum problema que cause a perda do registro de pagamento da apólice de seguro pelo usuário, não haverá efetivação da transação e o mesmo não estará com seu bem segurado. E, devido ao mecanismo utilizado para as operações realizadas, haverá latência de várias horas até a comunicação do fato, bem como dificuldade de

verificação para encontrar o ponto de falha, considerando o processo completo.

Além de que este modelo necessita de ações de *backoffice* para verificação do processamento das transações e efetivação dos contratos, identificando irregularidades e problemas, e atuando motivados por controle internos de conformidade ou por reclamações em canais de ouvidoria. No exemplo dado, caso haja alguma irregularidade como o cadastramento da pontuação de bônus do seguro do cliente de forma equivocada, apenas após o processamento da transações pelo nó da entidade reguladora esta questão será identificada para ser tratada em fluxo de exceção, gerando maior complexidade ao processo de comunicação e possibilidade de uma pendência que será resolvida somente após contato do cliente ou verificação pelo *backoffice* do vendedor ou do seu parceiro.

Assim, os principais problemas das soluções atuais de operações interbancárias são que elas funcionam utilizando soluções legadas em que cada nó participante da operação negocial possui sua base de dados centralizada, sendo a transparência da operação muito baixa visto que as regras de processamento são mantidas por cada unidade processadora e os processos de auditoria são eventuais e dependentes de ações individuais de cada instituição em verificar suas ações de processamento.

Outro problema é a latência na efetivação das transações, pois o cliente final só tem visibilidade do sucesso ou de problemas em suas transações muito depois da realização da mesma, devido a dependência dos processamentos em cada um dos nós participantes. Além de que, apesar da operacionalização neste contexto não ser complexa, ela é trabalhosa, sendo onerosos os gastos com processamento em lote e monitoração das *schedules* destes, estando a responsabilidade pulverizada entre os diferentes participantes e não havendo a possibilidade de manter uma visão única desse contexto.

Assim, no modelo atual a integração negocial entre as entidades não é refletida nos sistemas de comunicação das operações, sendo necessário processos de *backoffice* para identificação de situações em que haja falha de efetivação de transações, processamentos incompletos ou inconsistentes e ausência das respostas esperadas.

O modelo de negócio prevê compartilhamento transparente de responsabilidades entre os participantes, porém as soluções tecnológicas implementadas atualmente não acompanham este modelo de integração, sendo a *blockchain*, como *ledger* distribuído, uma ferramenta interessante para suprir esta necessidade.

3.1.2 *Blockchain* Como Solução

A *blockchain* é uma opção coerente e robusta neste roteiro de integração de fluxos de negócios financeiros compartilhados, em que disponibilidade e transparência são exigidas, assim como conformidade e auditoria. Os pontos de falhas são reduzidos com a implementação do processamento descentralizado, há diminuição dos custos pela adoção da operação compartilhada, além de celeridade na efetivação das transações por não depender de processamentos centralizados independentes ou das respectivas janelas de horário de *schedules* de execução e retornos assíncronos por interfaces baseadas em arquivos *batch*.

A arquitetura *blockchain* é uma alternativa às soluções tradicionais de operações interbancárias para que as instituições possam compartilhar bancos de dados distribuídos, ou seja, o livro-razão de determinados produtos. Desta forma, a *blockchain* pode substituir o processamento sequencial em lotes dos arquivos

de interfaces da soluções legadas, em que as entidades trocam arquivos como mecanismos de comunicação sobre as operações interbancárias, processando-os de forma centralizada.

Diferentemente, a *blockchain* prove um *ledger* distribuído em que as transações são processadas de forma descentralizada e *online* diminuindo a latência da efetivação das operações, além de prover um mecanismo para auditoria constante e independente, complementadas por abordagens de privacidade e confiança coerentes. Além da possibilidade de dispensa ou de renegociação dos termos de atuação da entidade verificadora, que não precisará ter a responsabilidade e o mandato únicos para promover a validação das operações realizadas considerando a confiança dos dados, pois os nós compartilham a solução e as informações necessárias para que esta verificação seja realizada por mecanismos da rede.

Em resumo, os diagramas 3.1 e 3.2 apresentam justamente a principal diferença entre as soluções legadas e a *blockchain* para processamento das operações interbancárias. Enquanto esta compartilha o *ledger* e permite processamento *online*, a outra depende da transferência de arquivos e dos processamentos em lote independentes e centralizados. O diagrama 3.1 apresenta o modelo utilizado nas soluções de operações interbancárias legadas e o diagrama 3.2 apresenta o modelo utilizando *blockchain*.

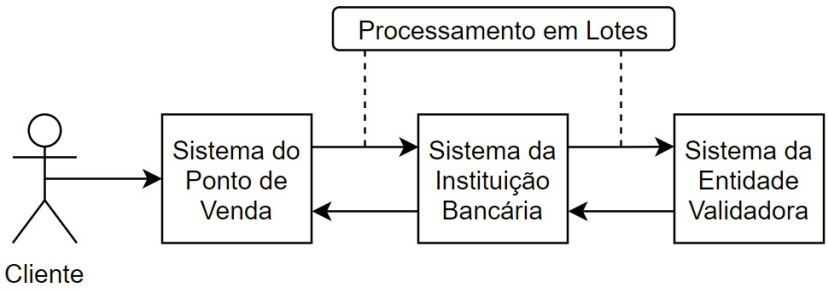

Figura 3.1: Cenário da solução legada

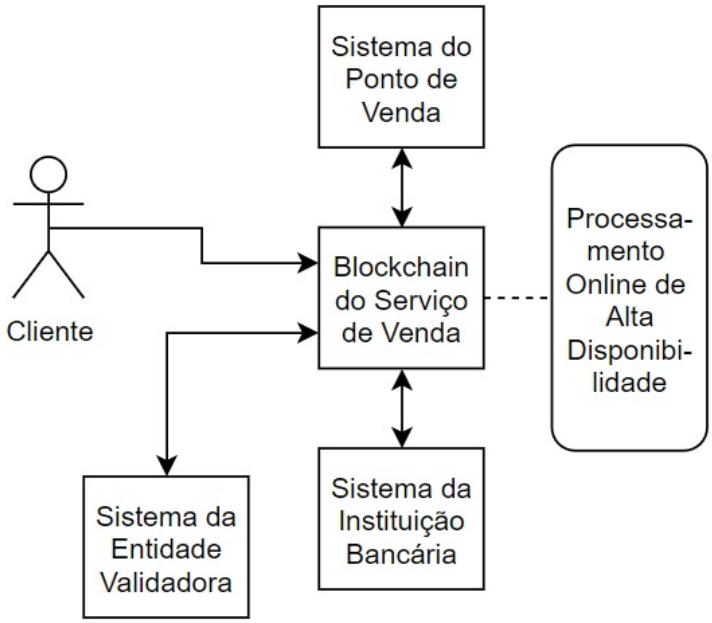

Figura 3.2: Cenário da solução *blockchain*

Além das características inerentes a um sistema distribuído que credenciam a *blockchain* como solução para comunicação de operações interbancárias, tais como descentralização e disponibilidade, esse ainda apresenta implementações para concordância entre os nós da rede com a intenção de solucionar o problema da confiança no contexto distribuído, entre as quais os algoritmos de consenso, as operações de criptografia e as regras de imutabilidade e transparência do *ledger*.

Nas operações interbancárias entre entidades financeira a questão da confiança é sensível, pois os participantes compartilham um fluxo de negócio em que cada um possui sua intenção específica e, apesar de estarem de acordo, não confiam entre si justamente pela independência de processamento das transações. Desta forma, cada um mantém seus processos internos de verificação das operações e dependem de autoridades externas para validação dos dados compartilhados, ou seja, os nós participantes do fluxo de negócio não confiam entre si e preferem replicar os esforços de processamento e operacionalização, além de bancar entes externos para auditar o processo.

Assim, a *blockchain* por exigir consenso da rede, possibilitar utilização de criptografia para proteção de acesso e privacidade dos dados e manter um banco de dados distribuído imutável promove confiança em sua rede descentralizada, que não dependerá de entidades externas para auditoria e superará o problema da falta de transparência dos processamentos centralizados. Mudando de um paradigma de latência na efetivação das transações e falta de confiança entre as partes para um contexto em que as transações são verificadas por uma solução compartilhada de forma transparente e sem a necessidade de validação por uma autoridade reguladora, a qual caso exista pode ser apenas comunicada sobre as operações ou acessar a *blockchain* para auditoria e monitoramento, ações estas que os próprios participantes da rede podem concordar em delegar a entidades que não participem do fluxo de negócio, mas tem papel de conformidade na rede a fim de evitar ações espúrias de grupos de nós mal intencionados atuando conjuntamente.

3.1.3 Trabalhos Correlatos

Neste contexto de novas alternativas de soluções tecnológicas para modernização dos processos bancários existem vários trabalhos em desenvolvimento com resultados interessantes, entre os quais relaciono na tabela 3.1 os mais relevantes a essa dissertação.

Tabela 3.1: Trabalhos Correlatos

Nome do Trabalho	Descrição
GT *Blockchain* Febraban	O grupo de trabalho *Blockchain* Febraban (Federação Brasileira de Bancos), formado por 18 bancos e instituições do setor, foi criado em 2017 para pesquisa e desenvolvimento em *blockchain* e apresentou no congresso CIAB 2018 um protótipo *blockchain* implementado no *Hyperledger Fabric* para compartilhar informações de segurança dos dispositivos móveis dos clientes que os utilizam para acesso aos serviços bancários, com o objetivo de alimentar sistemas antifraude com esses dados [50].

Plataforma PIER	A *blockchain* de consórcio PIER (Plataforma de Integração de Informações das Entidades Reguladoras) constituída pelo Banco Central (BC), a Comissão de Valores Mobiliários (CVM), a Superintendência de Seguros Privados (Susep) e a Superintendência Nacional de Previdência Complementar (Previc) começou a ser desenvolvida em agosto de 2017 como uma rede *peer-to-peer* para troca de dados entre esses órgãos reguladores do sistema financeiro [51].
Sistema Financeiro Digital	Anunciado no 11º Fórum Internacional de Tecnologia da Informação do Banrisul, o sistema SFD (Sistema Financeiro Digital) é um sistema *blockchain* consórcio desenvolvido pelos bancos Caixa Econômica Federal, Banco do Brasil, Sicoob, Banrisul e Santander para transferências de valores e pagamentos de forma simplificada utilizando aplicativo *mobile banking* ou aplicativo de celular específico pelos clientes dos bancos participantes [52].
Interbank Information Network	*Blockchain* do banco americano JP Morgan construída no módulo *Quorum* do *Ethereum*, que permite o compartilhamento de informações entre diversos bancos internacionais sobre transferência de valores entre fronteiras com o intuito de aplicar verificações de prevenção à lavagem de dinheiro pela implementação de *smart contracts* e processamentos descentralizados [53].
PIX	Apesar de não ser uma solução *blockchain* ou uma rede *peer-to-peer*, o sistema de pagamento instantâneo brasileiro (PIX) criado pelo Banco Central (BC) instituiu uma plataforma robusta para pagamentos ou transferências entre as 734 instituições credenciadas, sendo um exemplo aderente de solução de integração bancária com disponibilidade, escalabilidade e desempenho [54].
Open Finance	Outra iniciativa do Banco Central que mesmo não utilizando *blockchain* foi constituída com a intenção de integração de dados entre instituições financeiras, que no caso compartilham informações das movimentações de clientes de produtos e serviços financeiros a partir de diferentes plataformas conforme a autorização destes [55].

3.2 ARQUITETURA DO MODELO DE USO PROPOSTO EM *BLOCKCHAIN* PARA AS COMUNICAÇÕES DE OPERAÇÕES INTERBANCÁRIAS

Este trabalho propõe um modelo de utilização de *blockchain* para comunicação de operações interbancárias entre instituições financeiras, em que as premissas de escalabilidade, desempenho e privacidade têm fundamental importância na rede distribuída *peer-to-peer*, cujo os nós transacionam com disponibilidade, transparência e auditoria permanente.

O modelo tem governança de consórcio, em que os nós da rede são as pessoas jurídicas que compartilham fluxos de negócios cuja a contabilidade seja realizada na *blockchain*. E a arquitetura é delimitada em seis camadas, as quais sejam: a) camada de aplicação, b) camada de auditoria, c) camada de acesso, d) camada de consenso, e) camada de rede, e f) camada de dados. O diagrama 3.3 representa essa organização e elenca seus mecanismos.

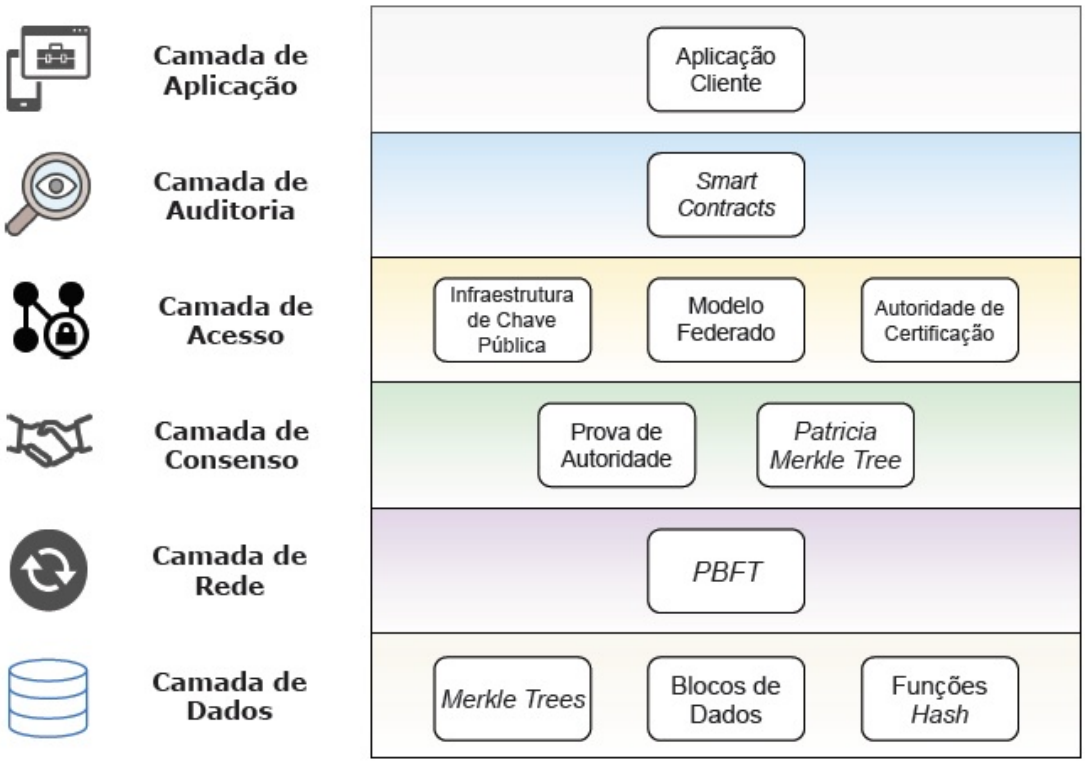

Figura 3.3: Diagrama da arquitetura do modelo proposto e seus componentes

A **camada de aplicação** estabelece os mecanismos para comunicação entre o *software* da entidade participante do consórcio e a *blockchain*. Como o modelo de governança é de consórcio, em que os participantes da rede são conhecidos e interagem negocialmente no mundo real anteriormente ao contato na rede *blockchain*, uma aplicação cliente deve ser compartilhada entre eles, para que cada um faça a instalação *on-premises* em sua infraestrutura de tecnologia, podendo ser o acesso à rede por *internet* ou *VPN*, a depender do acordo entre os participantes.

Na **camada de auditoria** ficam os componentes que executaram os processos de conformidade e monitoramento da rede para evitar que participantes legítimos do consórcio executem ou promovam acordos paralelos para a realização de ações fraudulentas. Nesta camada, os *smart contracts* desempenham pa-

pel fundamental, pois eles podem ser implementados na rede para execução desses processos de forma permanente e automática.

Já a **camada de acesso** estabelece os mecanismos para identificação dos participantes da rede e permissionamento, que serão fundamentais para definição das permissões nas camadas subsequentes. Para execução dos processos inerentes a esta camada a infraestrutura de chave pública, a autoridade de certificação e o modelo federado, considerando a governança de consórcio da rede, são implementados para segurança e privacidade no *blockchain*.

A **camada de consenso** é responsável pelo processo de identificação dos nós que participaram da verificação das transações para geração no bloco que poderá ser persistido na *blockchain*, sendo o algoritmo de prova de autoridade o mecanismo utilizado no modelo. Enquanto a **camada de rede** mantém o mecanismo de propagação do bloco para validação pelos nós da rede *peer-to-peer*, assim o algoritmo envolvido nesse contexto é o do *Practical Byzantine Fault Tolerance (PBFT)*. No modelo proposto as camadas de consenso e de rede foram bem delimitadas, pois os objetivos são diferentes e os participantes dos processos também, mas conjuntamente estabelecem um mecanismo de consenso híbrido na rede.

Por fim, a **camada de dados** é a responsável pela persistência dos blocos validados de forma encadeada, assim são mecanismos utilizados nesse contexto as operações de criptografia e as funções de *hash*. Esta camada é a responsável pela atualização do *ledger* e pelo mecanismo de encadeamento dos blocos para garantir imutabilidade.

3.2.1 Características do Modelo

A intenção do modelo proposto é prover uma rede distribuída *peer-to-peer* para comunicação de operações interbancárias por diversas instituições financeiras com garantia de escalabilidade, transparência, disponibilidade, desempenho, segurança e privacidade, eliminando a necessidade de um terceiro regulador e mantendo auditoria constante.

Como no contexto desta proposta de uso há um certo nível de confiança devido ao acordo jurídico preexistente entre as entidades para compartilhar um fluxo negocial visando lucro, a *blockchain* de consórcio é a melhor opção de governança por evitar a necessidade de onerosos mecanismos de consenso, como o algoritmo de prova de trabalho, e de processos de disputa ou de recompensa semelhantes aos *blockchains* públicos, inclusive tendo a liberdade de ser mais arbitrária na definição do papel dos nós.

Esta definição ajuda no desempenho e na escalabilidade da *blockchain*, assim como optar pelo *Patricia Merkle Tree* como estrutura de dados do cabeçalho do bloco invés do mecanismo mais comum que é o *Merkle Tree*, pois, apesar de ser uma estrutura mais complexa, aquele tem a vantagem de ter um processo de verificação mais veloz e gastar menos poder computacional para montar a estrutura de dados.

Auxilia também nesse contexto a opção pelo algoritmo de prova de autoridade para mecanismo de consenso, pois não exige alto gasto computacional para resolução de problemas matemáticos e se baseia na identidade dos participantes, algo de melhor desempenho e apropriado a uma *blockchain* de consórcio, principalmente no caso, em que há uma camada de controle de acesso para gestão das permissões dos nós da rede. De forma similar, a camada de rede com a utilização do protocolo de tolerância às falhas

bizantinas práticas apresenta ganho de desempenho e escalabilidade, pois o processo de validação não necessita passar pela maioria da rede, além de ser diligente na troca de mensagens por exigir retorno dos nós acionados [29]. Estabelecendo, assim, um modelo híbrido de consenso que atende às necessidades do contexto de comunicação das operações entre entidades bancárias.

A camada de acesso na arquitetura tem papel fundamental para realizar a identificação dos nós e estabelecer suas permissões, as quais estão condicionadas aos seus papéis na rede, assim esse credenciamento é utilizado pelo algoritmo de prova de autoridade para estabelecer os nós que montam a estrutura da *Patricia Merkle Tree* e definir quais fazem a validação. Auxiliando também em questões de privacidade e de sigilo na rede, definindo a permissão conforme o grau de anonimato especifico dos dados.

Assim, o controle de acesso tem papel fundamental para execução das diretrizes de privacidade do *blockchain*, porém o mandato de supervisão fica com a camada de auditoria, que monitora ações na rede, atua proativamente contra ações fraudulentas e executa protocolos para evitar centralização na rede. Este mandato não pode ser concedido a participante da rede que tome parte de fluxo de negócio mantido no *blockchain*, assim as entidades financeiras devem estar de acordo em manter a implementação de *smart contracts* que executem os processos inerentes à supervisão.

3.2.1.1 Utilização da Governança de *Blockchain* de Consórcio

O modelo de uso proposto nesse trabalho adota como arquétipo de governança a *blockchain* de consórcio, pois nessa configuração participam da rede *blockchain* um grupo de entidades parceiras com interesse de compartilhar o *ledger* de transações. Diferentemente do modelo público em que os participantes não se conhecem e não possuem acordos no mundo real visto a rede ser aberta, ou do paradigma privado em que apenas uma entidade mantém a rede, sendo ela constituída por uma pessoa ou um grupo com representação e mandato únicos.

Assim, a *blockchain* de consórcio é aderente à necessidade da rede de comunicação de operações interbancária, em que instituições financeiras estabelecem parcerias que refletem um processo de negócio compartilhado, cujas transações são registradas no *ledger blockchain* e os nós da rede são mantidos pelas entidades participantes do consórcio.

Outra questão é que a organização dos papéis na rede distribuída no modelo da *blockchain* consórcio é mais próximo do problema das operações interbancárias abordado no trabalho, pois neste desenho em um grupo de nós alguns têm permissão para ler a cadeia e outros não têm, assim a participação na atualização do *ledger* não é abrangente, mas é delegada aos nós previamente selecionados. Ou seja, a *blockchain* não fica atrelada a uma gestão centralizada, como no modelo privado, mas também não é aberto para qualquer participante que cumpra os requisitos do mecanismo, como na *blockchain* pública.

Desta forma, o modelo de consórcio atende a perspectiva do compartilhamento do *ledger* entre entidades parceiras em um fluxo de negócio específico, em que uma camada de autorização delimita o papel dos nós para atender diretrizes baseadas em regras de negócio que atendam a natureza dos acordos entre as entidades participantes da rede.

Além da *blockchain* de consórcio atender ao contexto da parceria entre as instituições financeiras para

compartilhar a comunicação de operações interbancárias em um *ledger* integrado, esse modelo permite estabelecer arranjos que auxiliam a garantir escalabilidade, desempenho e privacidade, pois este tipo de organização possui maior potencial para flexibilização de determinados aspectos em comparação às duas outras governanças, pública e privada.

Em comparação ao modelo de referência de *blockchain* mais comum, que é o de governança pública, o qual exige processos onerosos nas camadas de consenso e de rede dada sua natureza aberta, em que os nós disputam a seleção para participação no mecanismo de consenso e a validação deve considerar a verificação pela maioria da rede, o modelo de consórcio possui mecanismos mais simples, tendo menos prejuízo com relação a escalabilidade e ao desempenho.

Como o *blockchain* público é aberto e os papéis dos nós equânimes, em cada ciclo de verificação das transações a quantidade de nós que superam a prova para participar do mecanismo de consenso varia, assim como a resposta de verificação dos blocos, que depende do poder dos nós ativos naquele determinando ciclo. Assim, essa incerteza dificulta a escalabilidade e pode gerar gargalos inesperados.

Estes problemas são superados na *blockchain* de consórcio, em que não há necessidade de disputa para definição dos participantes do mecanismo de consenso e a verificação não precisa considerar a maioria da rede, pois os papéis dos nós dependem da definição de regras da rede considerando o fluxo de negócio atendido, assim há controle quanto aos participantes do consenso e da verificação de forma a manter a confiança na rede e otimizar o processo decisório, promovendo escalabilidade e reduzindo a possibilidade de obstáculos com relação ao *throughput* dos processos.

Outro ganho relacionado à possibilidade de identificação de permissões para os nós da rede, invés de considerar a igualdade entre os participantes, é a possibilidade de considerar essas credenciais para definições de segurança e privacidade, pois esta deixa de depender apenas de operações de *hash* criptográficos como na *blockchain* pública para utilizar além desses, outros mecanismos na camada de segurança.

A *blockchain* pública é mais segura conforme mais nós participem do consenso, porém isso aumenta o gasto de poder computacional ou de risco da prova utilizada, comprometendo o desempenho, sendo o inverso verdadeiro. Na *blockchain* de consórcio não ocorre este dilema, pois conforme a permissão do nó, esse desempenhará determinado papel na rede que, por possuir apenas participantes conhecidos, precisa somente que eles sejam autenticados.

No contexto destas questões, a comparação com a *blockchain* privada é mais distante, pois o problema da comunicação de operações interbancárias utilizando um *ledger* compartilhado não deve ser resolvido por esse tipo de governança, que necessita de uma entidade centralizada mantendo a rede, o que descredencia este tipo de *blockchain* como solução, pois um banco de dados centralizado mantido pelo "dono" do mecanismo, em que os dados fossem acessíveis por entidades parceiras, já resolve a situação. Além de que, tecnicamente, a comparação dependeria das diretrizes estabelecidas na *blockchain* privada específica e não das características inerentes ao modelo de governança.

3.2.1.2 Processo de Consenso e Validação do Bloco

Duas camadas neste modelo que se diferenciam bastante das escolhas comuns nas implementações de *blockchains* são as de consenso e de rede. Geralmente as *blockchains* utilizam como algoritmos de consenso a prova de trabalho ou a prova de risco, inclusive quando trabalham de forma híbrida, tendo no seu desenho de arquitetura a camada de consenso com a responsabilidade de executar o mecanismo de prova e a validação do bloco pela rede.

Diferentemente, o modelo de *blockchain* proposto nesse trabalho possui a peculiaridade de separar em duas camadas distintas o processo de obtenção de consenso sobre a geração de um bloco e o processo de validação do bloco pela rede. Esta independência permite que nós diferentes participem dos processos distintos, além de que possibilitar a execução dos processos paralelamente para blocos distintos sem o impacto de aumentar a frequência de *forks* na cadeia.

A intenção é estabelecer um mecanismo de consenso híbrido em que haja redução das ramificações na rede pela adoção de diretrizes que determinem melhor delimitação das rodadas de consenso, assim como no algoritmo de prova de risco. Bem como implementar artifício para que mesmo quando a quantidade de nós da rede seja maior que a necessária o algoritmo de prova de autoridade se mantenha consistente, evitando *forks* na cadeia. E determinando um número coerente para execução do mecanismo de tolerância às falhas bizantinas práticas, superando a necessidade de manter como participantes do consenso um número reduzido de nós.

Desta forma, como o controle sobre o consenso quanto às transações que comporão um bloco é realizado por determinado grupo de nós, eles têm o controle das transações que já foram processadas e validadas, assim não há retrabalho com a verificação repetida de transações e há controle dos blocos. A ação seguinte, que é a de validação dos blocos pela rede, é realizada em paralelo à primeira, porém por outros nós, e é realizada por estes de forma sequencial para evitar os *forks*.

Ou seja, as camadas trabalham em paralelo executando seus processos distintos de forma sequencial, com isso há ganho de desempenho, além de permitir melhorar a escalabilidade, que executará balanceamento quando houver disponibilidade de nós que possuam permissão de atuação em uma das duas camadas ou em ambas. A possibilidade de ocorrer enfileiramento no processo de validação dos blocos é baixa, pois o mecanismo de consenso é mais oneroso que o de validação devido aos processos criptográficos envolvidos, já que o primeiro é responsável pela atualização da árvore de *Merkle* e o segundo por verificar, o que sempre é um processo mais rápido.

Para o mecanismo de consenso é utilizado o algoritmo de prova de autoridade, em que participam os nós credenciados pelas autoridades de certificação. Assim, um nó da rede é definido randomicamente e será responsável pela geração de um bloco, desde que duas premissas sejam obedecidas: a) o nó esteja ativo, e b) o nó tenha permissão de participação na camada de consenso. Outro nó é definido também randomicamente para determinar qual o *range* de transações que participarão do bloco que será gerado, sendo que este nó também deve obedecer às premissas anteriores e considerar o último *range* definido como base. Sendo que o nó isento, escolhido pelo acaso, vai buscar o parâmetro da quantidade de transações do *range* em mecanismo mantido pela camada de auditoria. As informações sobre estas duas eleições mais atuais e sobre os *ranges* de transações de cada bloco devem estar disponíveis para todos os nós.

O nó responsável pela geração do bloco atualizará a árvore de *Merkle* com o *hash* das transações definidas para o bloco utilizando o *Patricia Merkle Tree*. A estrutura de dados do cabeçalho do bloco utilizada como base a partir da qual será feita a atualização é obtida consultando o bloco mais recente da *blockchain*. Terminando o processo é obtido o *timestamp* de geração do bloco e ele é liberado para validação da estrutura de dados de seu cabeçalho.

Utilizando o mecanismo *Patricia Merkle Tree* invés do tradicional *Merkle Tree* o modelo de *blockchain* proposto apresenta melhor escalabilidade, pois os nós participantes não precisam manter toda cadeia para verificação do mecanismo de consenso e conseguir validar as transações apropriadas em determinado bloco, como ocorre por exemplo no *Bitcoin*. No mecanismo proposto, para participar da geração do próximo bloco da cadeia bastará obter a raiz da *Patricia Merkle Tree* atualizada, que pela compactação do modelo e seu funcionamento é bem menor comparativamente, por exemplo, o arquivo de *download* da cadeia *Bitcoin* em um de seus projetos mais famosos, o *Bitcoin Core*, possui atualmente 200 Gb [56].

No processo de validação do bloco as transações são confrontadas com a árvore de *Merkle* atualizada para verificação se elas correspondem ao bloco gerado. Este processo de validação utiliza o protocolo *Practical Byzantine Fault Tolerance*, em que os nós participantes do mecanismo trocarão mensagens após o processo de verificação do bloco, emitindo uma resposta positiva ou negativa a depender da correspondência das transações com o bloco, ou seja, da verificação de que a atualização da árvore de *Merkle* é coerente com os *hashes* das transações.

Os nós participantes do processo são selecionados entre aqueles com permissão para execução do processo de validação, considerando uma quantidade arbitrária proporcional ao tamanho da rede com o objetivo de evitar que em redes muito grandes o processo de troca de mensagens se torne muito oneroso. Assim, para o cálculo do número de participantes será utilizada a *distribuição t de Student* [57] considerando o intervalo de confiança estabelecido por Jerzy Neyman em 1937 [58] em relação a quantidade de nós ativos da rede com permissão para validação, quando este for igual ou superior à 30 nós. O mecanismo chegará a um acordo após três rodadas de troca de mensagens, nas quais grupos distintos de nós participam do processo. A última rodada tendo resposta positiva registra o *timestamp* de validação do bloco [29].

Quando não há consenso sobre o bloco ele é descartado e suas transações desconsideradas, sendo um ponto positivo o fato da distribuição das transações pelos blocos ser controlada, pois um tempo de latência é definido para a confirmação das transações e quem comandou a mesma pode considera-la como rejeitada após este tempo, verificar alguma inconsistência e fazer novo comando, invés de aguardar indeterminadamente ou ter a transação no estado pendente permanentemente.

Pode haver implementação na camada de auditoria que mantenha a informação do tempo de latência definido para a *blockchain*, assim transações inválidas ou não processadas até este limite podem ser atualizadas nos clientes dos respectivos nós como não autorizadas. O nó responsável pela geração do bloco inválido ficará inativo por um número arbitrário de rodadas, devido à possibilidade da inconsistência ser causada por tentativa de ação fraudulenta na geração do bloco.

O acordo da rede habilita a camada de dados a realizar a persistência do bloco na *blockchain* encadeando as seguintes informações no cabeçalho do bloco:

- número sequencial do bloco;

- *hash* do identificador do nó responsável pela geração do bloco;
- *timestamp* de geração do bloco;
- *hash* do identificador do nó que definiu o *range* das transações do bloco;
- identificador da primeira transação;
- identificador da última transação;
- *timestamp* de validação do bloco;
- *hash* do cabeçalho do bloco anterior;
- *timestamp* corrente;
- *hash* do nó raiz da árvore de *Merkle* do protocolo *Patricia Merkle Tree*.

O bloco então é disponibilizado para atualização dos nós ativos da rede, sendo uma das premissas para se manter ativo ter o último bloco registrado na cadeia ou ter toda a cadeia atualizada, desde que a árvore de *Merkle* seja a mais recente.

A atualização da *blockchain* não obedece ao número sequencial do bloco, sendo esta informação utilizada para auditoria e monitoração. Porém, para garantir a imutabilidade da *blockchain*, não pode ser inserido um bloco com *timestamp* corrente menor do que o do último bloco da cadeia, além obedecer a regra geral de que qualquer alteração ou tentativa de alterar uma transação anterior será invalidada e rejeitada pela execução do mecanismo de consenso.

A definição quanto aos mecanismos apresentados tiveram como principal motivação obter desempenho e escalabilidade aproveitando o fato da *blockchain* proposta ter governança de consórcio, em que a responsabilidade de verificação das transações é dividida entre participantes que se conhecem e possuem um acordo prévio. Assim, o contexto favorece a utilização do algoritmo de prova de autoridade, que é bem menos oneroso, principalmente no modelo proposto em que entre os nós com autoridade a disputa é resolvida randomicamente, visto não haver especialização no contexto ou qualquer diferenciação a ser considerada.

A utilização do *Patricia Merkle Tree* fornece uma opção para verificação mais rápida que o *Merkle Tree* tradicional, facilitando as rodadas de validação com o *Practical Byzantine Fault Tolerance*, assim a latência no retorno quanto a consistência do bloco é menor, o que traz mais desempenho ao mecanismo. Esta qualidade ainda é aprimorada com a definição de um mecanismo simples para o controle de processamento de transações, que é a definição aleatória de um nó isento para informar o conjunto de transações que constituirá o bloco, conforme parâmetro estabelecido com base no monitoramento da rede. Assim, evita-se o retrabalho de blocos diferentes processando as mesmas transações e trabalha-se com otimização considerando o volume transacional da rede, obtendo ganhos de escalabilidade e desempenho ao fazer os nós trabalharem de forma mais ajustada.

A atuação conjunta das camadas definidas nessa arquitetura permite que os mecanismos compartilhem informações e trabalhem conjuntamente buscando a otimização dos processos. Desta forma, aproveitando

o contexto mais flexível da governança em consórcio e a existência de uma camada responsável pelo monitoramento da rede e por manter os dados a serem utilizados por mecanismos de outras camadas da *blockchain* para obter resultados em escalabilidade, desempenho, segurança e privacidade.

3.2.1.3 Diretrizes para Segurança e Privacidade

No modelo proposto a camada de segurança possui um mecanismo que funciona como uma autoridade de certificação, a qual credencia as permissões dos nós na rede, estabelecendo uma infraestrutura de chave pública mantida pela própria rede *peer-to-peer*. Assim, cada nó tem uma identificação comprovada no mundo real e uma chave pública associada, as mensagens na *blockchain* utilizam este contexto criptográfico assimétrico, em que as transações são assinadas com a chave privada do emissor utilizando uma função *hash SHA256*.

Para que a autoridade de certificação não seja um ponto de falha possivelmente explorado em ataques, essa deve ser distribuída na *blockchain*. Desta forma, a *blockchain* manterá entidades na rede com implementações de autoridades de certificação que atuarão conjuntamente aos nós e exercerão consenso para manutenção das diretrizes de permissionamento da rede. Estas autoridades de certificação devem ser monitoradas pela camada de auditoria para garantir a atualidade das regras de permissão executadas.

As chaves privadas impedem que os outros, além do detentor do *token* de acesso, acessem os dados de determinado contexto, porém para um fluxo de negócios funcionar algumas informações devem ser compartilhadas entre os parceiros e outras ter o sigilo mantido por criptografia privada. Assim, no modelo proposto em alguns contextos são utilizados algoritmos de chaves simétricas, em que o *token* de acesso é compartilhado entre os nós envolvidos no fluxo de negócio.

Este é um diferencial do modelo proposto, que divide a estrutura de dados das mensagens para trabalhar com ambos os contextos, simétrico e assimétrico simultaneamente, para que se obedeça o grau de anonimato para cada registro da *blockchain*, ou seja, especifica-se quem pode acessar o quê, assim as informações só estarão disponíveis aos nós competentes conforme validação de suas credenciais.

A camada de segurança atua em um modelo federado em que os próprios nós mantém o mecanismo da autoridade de certificação, visto que todos são responsáveis pela *blockchain* e pelos fluxos de negócios ali contidos. Assim, cada nó participante da rede é identificado por sua chave pública e detém sua chave privada que permitem o acesso a determinados dados, desta forma, uma transação gerada por determinado nó é encriptada com a chave privada deste, mas um *token* de acesso que pode ser cedido pelo próprio nó para acesso aos dados por outro nó, em um contexto de confiança baseado na federação.

Desta forma, os dados são compartilhados entre os nós que podem acessá-lo, sendo a autoridade de certificação estabelecida na *blockchain* a entidade que mantém as permissões de acesso. Assim, caso determinado nó precise acessar certo dado da rede, esse deve solicitar autorização à autoridade de certificação, que cederá um *token* válido de acesso ao dado, desde que o nó tenha permissão de consumo da informação pelo consórcio. Os nós participantes podem atualizar as prerrogativas de permissão de seus dados aos demais nós junto à autoridade de certificação, cedendo ou retirando permissão de acesso aos dados relacionados a determinado fluxo de negócio.

Este mecanismo é do tipo de privacidade de dados adicional por fornecer uma camada de controle de acesso aos dados mantida pela rede *peer-to-peer*, que mantém a lista de controle de acesso pelo relacionamento entre os nós participantes da rede, respondendo a um contexto em que usuários autorizados acessam aos dados permitidos e técnicas de criptografia impedem que usuários sem autorização acessem.

O modelo proposto ao mesclar os mecanismos de criptografia simétrica e de criptografia assimétrica possui o benefício de permitir o compartilhamento de informações por parceiros de negócio que participam de fluxo de atendimento comum, de forma a ter segurança e privacidade garantidas, aproveitando o modelo federado para exercer a autenticação de forma descentralizada, em que o dono da informação legitima a permissão de acesso aos dados.

Os mecanismos propostos para a camada de segurança permitem que os nós participantes da rede *blockchain* delimitem o que pode ser acessado e o que não pode ser acessado por parceiros ou não parceiros do fluxo de negócio específico. Assim, o modelo permite que o *layout* de uma transação negocial completa seja divido em várias transações na operação da *blockchain* para delimitar permissões distintas.

Por exemplo, uma transação de financeira de transferência de valores com as seguintes informações: banco de origem, conta de origem, banco de destino, conta de destino, documento de identificação do emissor e valor da operação, pode ser dividida em duas transações: a) banco de origem + banco de destino + valores e b) conta de origem + conta de destino + documento de identificação do emissor. Desta forma, a primeira transação poderia ser pública entre todos os parceiros para fins de transparência contábil, enquanto a permissão de leitura da segunda transação seria apenas para os bancos envolvidos na operação, garantindo privacidade e sigilo bancário.

Esta estrutura é fundamental no contexto das operações interbancárias, em que questões como a lei geral de proteção aos dados e o sigilo bancário são tão importantes. Assim, o mecanismo possibilita em um contexto descentralizado de compartilhamento de dados garantir privacidade e proteção. Sendo fundamental para manter a confiança entre os parceiros, pois os dados estão protegidos e são compartilhados apenas com a autorização do "dono" da informação.

3.2.2 Processo de Auditoria

A camada de auditoria é outro mecanismo do modelo proposto que não está nas referências de *blockchain* mais comuns, mas desempenha um papel fundamental, principalmente considerando o delicado contexto das operações interbancárias, que, em geral, tratam de transferências de valores, bens e direitos, devendo prezar pelo sigilo e conformidade. Desta forma, a camada de auditoria desempenha o papel de monitoramento da rede para garantir que os acordos entre os nós participantes sejam respeitados.

Nesse exercício de monitoramento da *blockchain* a camada de auditoria é responsável por manter alguns parâmetros comuns aos nós da rede que são utilizados nos mecanismos de outras camadas, como: a) a quantidade de transações de um bloco, b) o tempo de vida das transações e c) a identificação do nó como validador, verificador ou ambos.

A primeira informação é utilizada para otimizar o processamento de transações pela *blockchain*. Já o segundo parâmetro permite que o nó atualize o status de sua transação na camada cliente de pendente para

inválida, caso ela não seja processada no limite de tempo estabelecido no parâmetro ou esteja em um bloco não validado. E o terceiro é utilizado nas camadas de consenso e de rede na seleção dos nós responsáveis pela prova de autoridade e pelo protocolo de tolerância às falhas bizantinas práticas, que funcionam como um modelo híbrido na proposta.

O parâmetro da quantidade de transações de um bloco é fixo e será estabelecido pelos parceiros de negócio considerando as estimativas do fluxo de atendimento ao produto na *blockchain* e a quantidade de entidades participantes desta. Assim, havendo poucos nós na rede e muitas transações, a quantidade de transações em um bloco deve ser maior, e isso deve ser considerado para esta definição arbitrária do consórcio.

O segundo parâmetro de limite de tempo para validação da transação também considera a característica do fluxo de negócio atendido, em que necessitando de uma resposta rápida, menor o tempo limite do ciclo de vida da transação, caso contrário, um valor limite mais dilatado pode ser estabelecido, sem prejuízo, pois a velocidade de processamento da *blockchain* não depende desse parâmetro.

Já o terceiro parâmetro de identificação dos nós participantes dos processos da camada de consenso e da camada de rede, deve considerar o papel da entidade que mantém o nó no fluxo de negócio, considerando os riscos assumidos, de imagem e financeiro, para determinar se o participante pode ou não participar dos processos, conforme o fluxo de negócio atendido na *blockchain*.

Os dados do cabeçalho do bloco são importantes para o monitoramento da rede *blockchain*, principalmente as informações: a) do nó responsável pela geração do bloco, b) do nó que definiu o *range* das transações do bloco, c) da identificação das transações limites do bloco, e d) do identificador do bloco. Com essas informações, mecanismos automáticos podem identificar:

- nós que estejam realizando ações não autorizadas, garantindo que as escolhas aleatórias não estão sendo fraudadas;
- nós que estejam gerando blocos inconsistentes;
- se a sequência das transações está sendo obedecida e que não há transações ficando sem processamento;
- frequências suspeitas nas definições randômicas do responsável pela geração dos blocos, ou seja, um nó ser escolhido numa frequência muito maior de dos demais.

Os mecanismos expostos na camada de auditoria promovem segurança e governança à *blockchain*, garantindo que as ações estejam de acordo com a implementação da rede e que os parâmetros sejam adaptáveis considerando o fluxo de negócio atendido, conforme acordo entre os parceiros.

Assim, a transparência é ampliada e ações de otimização da rede são menos onerosas e mais frequentes. O acordo e a intenção de compartilhamento de dados entre as entidades permite que os parâmetros de governança sejam estabelecidos em conjunto e mantidos por decisão colegiada dos parceiros para posteriormente serem ajustados na camada de auditoria, melhorando a gestão da rede.

3.2.2.1 Utilização dos *Smart Contracts* para Auditoria

As ações de monitoramento da *blockchain* serão mantidas na camada de auditoria pela implementação de *smart contracts* para que as execuções sejam automáticas e permanentes. Assim, os seguintes mecanismos serão implementados no modelo proposto: a) auditoria das permissões dos nós, b) verificação de integridade, c) controle de processamento de transações, d) monitoramento de comportamentos suspeitos e e) garantia de atualização das autoridades de certificação da rede. Com a implementação dos *smart contracts*, a *blockchain* manterá as ações de auditoria e conformidade sem a necessidade de uma terceira parte interveniente.

- **Auditoria das permissões dos nós.** Os blocos da *blockchain* serão verificados para identificar que o *hash* do identificador do nó responsável pela geração de um bloco é o de algum nó que esteja na lista de nós autorizados para executar esta ação. Sendo que as permissões dos nós são mantidas na camada de auditoria, conforme acordo entre as entidades parceiras estabelecendo o perfil de cada nó.

- **Verificação de integridade.** *Smart contract* da *blockchain* proposta é responsável por identificar entre os blocos inválidos quais foram os nós que os geraram, para determinar se este estão realizando ações inconsistentes na rede. Caso a frequência de blocos inválidos gerados pelo nó seja superior à 10% dos blocos disponibilizados por ele para verificação, o nó perde sua permissão, devendo negociá-la novamente com o consórcio. Ou seja, há uma tolerância razoável de 1 em cada 10 bloco gerados pelo nó ser invalidado pela rede, que se extrapolada gera penalidade ao nó.

- **Controle de processamento de transações.** A camada de auditoria verifica automaticamente se os nós estão respeitando a regra de apropriar as transações para verificação de forma sequencial. Assim, o *smart contract* compara o identificador da primeira transação e o identificador da última transação do cabeçalho dos blocos e verifica se as sequencias ausentes são em conjuntos de transações pertencentes aos blocos inválidos, e, caso não sejam, um alerta é emitido aos nós.

- **Monitoramento de comportamentos suspeitos.** *Smart contract* deve ser implementado para identificar se algum nó está participando com uma frequência muito alto do processo de geração do bloco, com o intuito de identificar possíveis fraudes no processo de definição dos responsáveis pela execução. Assim, caso um nó participe de um percentual maior que os definidos no escalonamento apresentado no quadro 3.4, ele terá as permissões na rede suspensas, devendo regularizar com o consórcio.

- **Garantia de atualização das autoridades de certificação da rede.** A camada de auditoria utilizará *smart contract* para verificar frequentemente o exercício de consenso e atualização das autoridades de certificação mantidas na rede, com o objetivo de que as regras de permissionamento sejam exercidas corretamente e com atualidade, além de identificar possíveis tentativas de ataques às autoridades de certificação visando ações fraudulentas ou inconsistentes quanto às permissões de determinados nós.

Os escalonamentos podem ser redefinidos pelos parceiros ou melhor ajustados conforme a característica da rede com relação ao número de nós participantes e o volume de transações do fluxo de negócio,

Quantidade de Nós Ativos na Rede	Percentual Definido
4 à 10	60%
11 à 50	40%
51 à 500	20%
> 501	10%

Figura 3.4: Quadro dos valores percentuais arbitrários definidos para o modelo proposto

sendo que a rede deve ter no mínimo 4 nós para atender premissa estabelecida no problema dos generais bizantinos [13]. O consórcio pode determinar que os parceiros podem constituir mais de um nó na rede, de forma igualitária ou proporcional ao papel no fluxo de negócio, com o intuito de deixar os mecanismos mais fortes devido à participação de uma quantidade maior de nós.

3.3 VANTAGENS PREVISTAS COM A APLICAÇÃO DO MODELO DE USO PROPOSTO

O modelo proposto permite a substituição das tradicionais soluções de comunicação de operações interbancárias por um banco de dados distribuído *peer-to-peer* de forma compartilhada, com mecanismos para garantir escalabilidade, desempenho e privacidade no contexto do escopo atendido e promove a confiança entre as entidades pela transparência, auditoria e monitoramento. Assim, os riscos da integração de dados entre diferentes entidades bancárias é reduzido, sendo possível inclusive não haver participação de uma terceira parte supervisora dos resultados do fluxo de negócio atendido.

Além de que uma solução compartilhada, como o modelo proposto, permite economia em infraestrutura de *hardware*, reduz os pontos de falha do sistema, aumenta significativamente a velocidade de comunicação, diminui gastos e esforços com operação, melhora disponibilidade e confiabilidade e facilita a escalabilidade. Desta forma, no contexto das operações interbancárias, há ganho para os parceiros de negócio e também para os clientes finais, que terão um fluxo de negócio de processamento mais rápido e a possibilidade de serviços mais baratos.

Ao prestar um serviço a um conjunto de parceiros é fundamental se preocupar com a velocidade de resposta e a disponibilidade do serviço, assim a *blockchain* proposta aproveita as características inerentes da sua governança de consórcio para utilizar mecanismos de consenso e de estruturação dos dados que permitam ganhos de desempenho e de escalabilidade, evitando os gargalos encontrados nos principais modelos de referência de redes *blockchain*, como o *Bitcoin* [1].

O modelo investe também em mecanismos de segurança, privacidade e auditoria que são fundamentais às solução bancárias compartilhadas, as quais devem prezar pela conformidade, pelo sigilo bancário e

pelo controle interno e externo. Tendo o consórcio de entidades parceiras seu papéis e responsabilidades definidos, com a garantia de que a rede monitora ações fraudulentas de forma autoexecutável.

O modelo proposto atinge seu objetivo ao trazer as vantagens das soluções distribuídas para substituir o processo legado das comunicações de operações interbancárias, estabelecendo um modelo que garanta transparência, autonomia no monitoramento e na verificação, disponibilidade, rastreabilidade e colaboração. Além de especificar mecanismos para melhor escalabilidade, desempenho e privacidade, atendendo às necessidades inerentes do negócio bancário.

4 IMPLEMENTAÇÕES E DISCUSSÕES DOS RESULTADOS

Para validar a proposta do modelo de comunicação de operações interbancárias utilizando *blockchain*, conforme descrito nesta dissertação, bem como estabelecer parâmetros de avaliação das principais definições do modelo, foram desenvolvidas provas de conceitos das propostas para demonstrar as funcionalidades e obter métricas de funcionamento com a intenção de corroborar as qualidades defendidas do modelo, que para fins de apresentação dos resultados foi nomeado *Bankchain*.

Assim, para efeito de validação do modelo *Bankchain*, três módulos foram desenvolvidos com objetivos delimitados, quais sejam: a) demonstrar a eficiência da utilização do *Patricia Merkle Tree* no mecanismo de encadeamento dos blocos, b) constituir o mecanismo *blockchain* de consenso híbrido e validação propostos no modelo, e c) implementar um algoritmo de criptomoeda, nomeado *Brucecoin*, semelhante ao *Bitcoin* [1] baseado na validação pela prova de trabalho para fins comparativos dos resultados com o *Bankchain*.

O objetivo do capítulo é comprovar as qualidades do modelo proposto considerando o que ele traz de diferente das implementações mais comuns de *blockchain*, demonstrando que o *Bankchain* é uma solução alternativa aos mecanismos tradicionais de comunicação de operações interbancárias por estabelecer um modelo baseado em um banco de dados distribuído *peer-to-peer*, como a maioria das *blockchains*, porém constituindo mecanismos que otimizem escalabilidade e desempenho, quando comparados a soluções de *blockchain* mais tradicionais, como o *Bitcoin*.

Este capítulo também descreve o ambiente de desenvolvimento de *software* usado na construção dos protótipos e detalha os cenários de implementação e de validação, discutindo as validações obtidas nos testes.

4.1 FERRAMENTAS E AMBIENTE DE DESENVOLVIMENTO

Para atender aos requisitos necessários à comprovação das qualidades dos mecanismos propostos no modelo *Bankchain* foi necessário a implementação de protótipos da implementação dos mecanismos mais singulares definidos na arquitetura de comunicação para operações interbancárias utilizando mecanismos do modelo *blockchain* proposto nesta dissertação.

Sendo fundamental neste contexto que os protótipos fossem módulos individuais que tornassem clara as vantagens da proposta e suas qualidades, tivessem objetivos bem delimitados e que os cenários de execução deixassem evidentes os benefícios em comparação aos mecanismos mais tradicionais de *blockchain*.

Os protótipos das funcionalidades do *Bankchain* foram construídos utilizando a linguagem de programação *Python* e alguns de seus módulos em métodos específicos, principalmente envolvendo criptografia. Com incorporação do módulo *Flask* para construção de aplicações *web* e utilização de ferramentas de desenvolvimento como *Postman*, *VS Code* e *Python IDLE 3.7* para construção dos cenários de testes com a

intenção de evidenciar os resultados das implementações propostas no modelo de solução apresentado na dissertação.

A tabela 4.1 apresenta um resumo das ferramentas utilizadas para a construção dos protótipos e as respectivas justificativas e motivações de adoção para o trabalho.

Tabela 4.1: Ferramentas Utilizadas

Ferramenta	**Características**	**Motivação**
Python Versão 3.7	Linguagem de programação de alto nível, orientada a objeto, interpretada e *procedural* de governança comunitária e aberta, além de ser uma linguagem livre e multiplataforma, extremamente portável.	Esta linguagem traz alta produtividade e fácil legibilidade, sendo bastante adequada para efeito da validação desta dissertação pela simplicidade e pela possibilidade de desenvolvimento de ferramentas interativas em contexto *procedural*.
VS Code	Editor de código-fonte da Microsoft com suporte para os sistemas operacionais *Windows*, *Linux* e *MacOS*, e realce de sintaxe e referenciamento automático.	Entre as principais características do *Visual Studio Code* temos: a) a disponibilidade de instalação da maioria das bibliotecas científicas comumente usadas, b) não ser necessário privilégios elevados para instalação de pacotes, c) gerenciamento intuitivo e fácil de pacotes pré-compilados, e d) ser multiplataforma.
Flask	*Micro-framework web* bastante simples, escrito em *Python*, utilizado para criação de microsserviços, expostos em *APIs RESTful*. Ele é expansível e permite que o projeto possua apenas os recursos necessários para sua execução.	Sendo as principais características do *Flask*: a) a simplicidade e leveza b) a baixa necessidade de configuração e suave curva de aprendizado, c) a arquitetura simples, d) a ótima expansibilidade, que permite ir incorporando novas necessidades à solução de maneira específica e contida, e e) a integração com o *Python*.

JSON	*JavaScript Object Notation* é um padrão para armazenamento e transmissão de dados em formato texto completamente independente de linguagem, sendo constituído por duas estruturas: uma coleção de pares nome x valor e uma lista ordenada de valores.	Modelo bastante difundido e de alta aceitação, tanto comercial, quanto acadêmica, devido à simplicidade, legibilidade, facilidade no *parsing*, tipagem, desempenho e compactação.
Postman	É uma aplicação que oferece serviço gratuito para testar *endpoints* e *APIs*, provendo uma interface gráfica para o usuário executar *APIs requests*, permitindo o recebimento e provimento de dados através de requisições.	O *Postman* provê um ambiente completo de desenvolvimento de *API* para projetar, simular, depurar, testar, documentar, monitorar e publicar elas, através de uma interface bastante simples.
Python IDLE 3.7	*IDLE* (*Integrated Development and Learning Environment*) é um ambiente de desenvolvimento integrado para execução de *scripts Python* lançado à cada nova versão da ferramenta.	As principais características são: a) fornecer um editor de textos em várias janelas com destaque de sintaxe, autocompletar, indentação rápida e outras funcionalidades, b) estabelecer um *Shell Python* com destaque de sintaxe, e c) possuir um depurador integrado interativo.

4.2 PROVAS DE CONCEITOS

O desenvolvimento dos protótipos apresentados neste trabalho foram baseados nas propostas do modelo de comunicação para operações interbancárias com *blockchain* conforme especificado nos itens 3.2 e 3.2.2, utilizando as ferramentas descritas no item 4.1, de maneira a resultar em módulos de sistemas, que embora não tenham acabamento e objetivo comercial atendem ao objetivo de demonstrar o funcionamento e as características das propostas do modelo *Bankchain*, por meio dos testes de validação em importantes cenários, os quais são descritos mais adiante neste capítulo.

O desenvolvimento reuniu métodos e padrões da engenharia de *software*, como programação orientada a objetos e utilização de tecnologias de redes para sistemas distribuídos, garantindo a decomposição modular, a abstração de dados e de serviços e a organização integrada. Sendo que as decisões de desenvolvimento foram pautadas no objetivo do ambiente de testes permitir a apresentação dos resultados de validação da proposta *Bankchain*. Comprovando que as abordagens propostas garantem escalabilidade e desempenho, contribuindo nas iniciativas de utilização de *blockchain* em modelos de negócios tradicionais, como é o caso da comunicação de operações interbancárias.

Com relação ao *hardware*, o ambiente de validação utilizado para as simulações do trabalho é composto por 1 computador do tipo *desktop* com Processador *Intel(R) Core(TM)-I7 3632QM 2.20 GHz* de *CPU*, com *4 Gb* de memória *RAM*, com um espaço em disco de *1 Tb*, responsável pelo ambiente de codificação e de testes. Já com relação ao *software* foi utilizado como sistema operacional do ambiente de desenvolvimento o *Windows 10*, além das ferramentas descritas no item 4.1.

4.2.1 *Patricia Merkle Tree*

O primeiro protótipo comprova a proposição estabelecida no modelo de comunicação de operações interbancárias *Bankchain* de utilização do *Patricia Merkle Tree* em substituição ao *Merkle Tree* usado na arquitetura de solução do *Bitcoin*. Assim, cumpre-se o objetivo de implementar um mecanismo em que para a validação do encadeamento dos blocos não seja necessário o *download* da cadeia completa. Resolvendo assim um problema bastante discutido do *blockchain Bitcoin*, conforme abordado no item 2.5, garantido assim melhor escalabilidade.

Uma das questões fundamentais que o modelo de comunicação de operações interbancárias utilizando a *blockchain Bankchain* estabelece é que o mecanismo do *hash* de referência do encadeamento dos blocos usado para validação da cadeia independe da necessidade de cada nó participante da rede fazer o *download* de toda a cadeia, pois ao invés de trabalhar com o algoritmo *Merkle Tree* em que a árvore é uma pirâmide em que todas as folhas devem ser conhecidas, o *Bankchain* utiliza o *Patricia Merkle Tree*, que funciona como um índice que mantém as folhas estruturadas de forma mais otimizada e orgânica.

Assim, o mecanismo desenvolvido neste protótipo ajudará na comprovação dos seguintes objetivos:

- apropriar a lista de transações que irão compor o bloco e gerar ou atualizar a raiz *hash* utilizando o mecanismo *Patricia Merkle Tree*;
- garantir que independentemente da ordem das transações o *hash* é o mesmo;
- realizar a validação do *hash* comprovando a independência da necessidade de recuperação da cadeia completa;
- comprovar a otimização quanto à armazenamento de dados.

O protótipo estabelecido no código *mpt.py* foi desenvolvido em *Python 3* utilizando suas bibliotecas: a) *eth-hash* para função de *hashing SHA-3*, b) *pycryptodome* para operações criptográficas, c) *rlp* para executar *Recursive Length Prefix* na serialização de objetos, e d) *cytoolz* e *toolz* para executar funcionalidades da

linguagem. Para o protótipo não houve necessidade de descentralização e o mesmo é *procedural*, visto que no modelo proposto a chamada é sequencial e controlada, realizada pelo nó com permissão no momento.

4.2.2 Consenso Híbrido

O segundo protótipo possui o objetivo de validar o mecanismo híbrido de consenso proposto no modelo *Bankchain* pela comparação do desempenho dessa proposta com uma implementação que adote as abordagens estabelecidas pelo *Bitcoin*, visto esta ser a estrututura de *blockchain* de referência, conforme discutido no 2.5. Assim, foram implementados dois algoritmos com o mesmo escopo, conforme estrutura apresentada na tabela 4.2, porém com diretrizes de funcionamento diferente com relação ao mecanismo de consenso.

Tabela 4.2: Escopo de Implementação das *Blockchains*

Função	Descrição
get chain	Recuperar a cadeia de dados da rede
connect node	Conectar os nós participantes da rede
add transaction	Adicionar transações para constituir o próximo bloco da rede
mine block	Minerar o bloco das transações disponíveis
replace chain	Executar consenso sobre a cadeia a ser mantida pela rede
is valid	Validar a cadeia mantida pela rede

O primeiro algoritmo *bankchain.py* adota a abordagem apresentada em 3.2.1.2 e detalhada nesta seção, em que os processos de concordância sobre a geração de um bloco e o processo de validação pela rede são distintos, porém complementares, no qual as camadas trabalham em paralelo executando seus processos de forma sequencial.

Já o segundo algoritmo *brucecoin.py* tem como referência de funcionamento o *Bitcoin*, assim estabelece uma criptomoeda conforme os parâmetros expostos nos itens 2.5 e 2.6.1, em que a mineração é conduzida pela realização do mecanismo de prova de trabalho, sendo o consenso estabelecido apenas pela comparação do tamanho da cadeia, vencendo sempre a de maior extensão.

O modelo de comunicação de operações interbancárias utilizando *blockchain* estabelecido com o *Bankchain* possui um mecanismo híbrido de consenso para persistência do bloco na cadeia. Assim, para comprovação de que as propostas estabelecidas no modelo são válidas foi necessária a construção de uma rede *blockchain* com as características estabelecidas para o mecanismo de consenso proposto.

Entre as características implementadas no protótipo para validação das propostas do mecanismo híbrido estão:

- definir randomicamente o nó responsável pela geração do bloco, desde que o nó esteja ativo e o nó tenha permissão de participar da camada de consenso;
- controlar o *range* de transações do bloco gerado, considerando o último *range* e identificando o nó responsável pela ação;
- selecionar entre os nós com permissão de validação da cadeia os que executarão o processo;

- implementar o mecanismo de interação para consenso através da conferências de valores e validação por três nós, em três rodadas independentes;
- montar o bloco gerado estruturado com as informações conforme proposto para o *Bankchain*.

No protótipo um método específico faz a definição do nó responsável pela geração do bloco, considerando informações simuladas, pois não cabe ao escopo de validação a persistência e gerenciamento do cadastro dos participantes da rede. Da mesma forma, um método específico controla o *range* de transações. Assim, para efeito de comprovação da proposta as informações que seriam mantidas pelo módulo de gerenciamento do cadastro e das ações dos participantes da rede serão simuladas, visto que estas não são objetos de comprovações das propostas.

Duas outras implementações fundamentais do protótipo da *blockchain*, conforme proposta nesta dissertação, são os métodos para seleção dos nós participantes do mecanismo de validação e o processo em si, também fazem parte do algoritmo *Bankchain*. Assim, conforme proposto, a validação é estabelecida em três rodadas com três nós distintos participantes, sendo o resultado positivo, o *timestamp* de validação do bloco é registrado.

O cálculo do número de participantes do processo de validação conforme a *distribuição t de Student* [57] não foi implementado, pois este somente faz sentido no contexto da existência de um módulo gerenciador do cadastro dos nós, que está fora do escopo do protótipo para os objetivos de validação propostos.

Outro método desta implementação é estabelecer o bloco com a seguinte estruturação de informações: a) número sequencial do bloco; b) *hash* do identificador do nó responsável pela geração do bloco; c) *timestamp* de geração do bloco; d) *hash* do identificador do nó que controla o *range* das transações do bloco; e) identificador da primeira transação; f) identificador da última transação; g) *timestamp* de validação do bloco; h) *hash* do cabeçalho do bloco anterior; i) *timestamp* corrente; e j) *hash* do nó raiz da árvore de *Merkle* do protocolo *Patricia Merkle Tree*. Obedecendo a regra de que não pode ser inserido um bloco com *timestamp* corrente menor do que o do último bloco da cadeia.

As implementações descritas comprovam o mecanismo de consenso híbrido proposto para o *Bankchain* e no cenário de teste desenvolvido no item 4.3.2 seu desempenho é mensurado para comparação com relação aos resultados de tempo de resposta entre esta implementação e o *blockchain* com mecanismos mais próximos da solução *Bitcoin*, o *Brucecoin*.

4.2.3 Considerações sobre Privacidade e Auditoria

Como os mecanismos de segurança e privacidade propostos no modelo *Bankchain* em 3.2.1.3, com destaque para as unidade de certificação descentralizadas, o modelo de federação e a divisão do *layout* de transações para privacidade específica permissionada são padrões e abordagens de mercado reconhecidos, não foram objeto de desenvolvimento de protótipo específico, pois não há ganho no contexto da validação as implementações de abordagens consolidadas.

Assim como, considerando a natureza da atividade de prototipação delimitada à comprovação dos benefícios dos mecanismos mais particulares da proposta *Bankchain* e a não utilização de *containers* de *IaaS* (*Infrastructure as a Service*) para não deturpar os resultados dos testes, não fez parte do escopo do protó-

tipo o módulo de gerenciamento do cadastro e das ações dos participantes da rede. Consequentemente, não faz parte deste trabalho a implementação dos *smart contracts*, visto a necessidade de uso de alguma plataforma proprietária, como *Hyperledger* [32], além do pré-requisito da disponibilidade dos dados mantidos por módulo de gestão dos participantes, conforme proposta em 3.2.2, e por ser uma abordagem também consolidada e padrão no mercado.

Em conclusão, para comprovação dos objetivos dessa dissertação não faz parte os escopos de privacidade e auditoria, apesar de suas estruturas fazerem parte da proposta de arquitetura da seção 3.2, detalhadas em 3.2.1.3 e 3.2.2, estas questões não são explorados nos cenários do item 4.3, que foca no objetivo de comprovação de escalabilidade e desempenho pela adoção do *Patricia Merkle Tree*, conforme 4.2.2, e do consenso híbrido, detalhado em 4.2.1, respectivamente.

4.3 DESCRIÇÃO GERAL DOS CENÁRIOS DE TESTES

Nesta seção são estabelecidas as evidências das validações obtidas com a execução dos protótipos desenvolvidos, considerando seus objetivos de verificação das propostas do modelo *Bankchain*.

Assim, os cenários de testes são explicados e são exploradas as motivações para a construção destes, bem como a descrição da execução do mesmo utilizando os protótipos desenvolvidos e explicando as vantagens obtidas com as abordagens propostas.

4.3.1 Cenário 1: Escalabilidade com *Patricia Merkle Tree*

Uma das questões estabelecidas como objetivo do modelo de comunicação de operações interbancárias proposto nesta dissertação é a utilização do *Patricia Merkle Tree* em substituição ao modelo *Merkle Tree* adotado no *Bitcoin*. Sendo que esta abordagem é defendida com a intenção de obter melhor escalabilidade na arquitetura proposta ao superar questões conhecidas, que são problemas do *Bitcoin*, conforme apresentado no item 2.5.

Quatro comprovações foram trabalhadas para cumprir as evidências de escalabilidade da proposta, conforme:

- **primeira validação:** capacidade de reunir um grupo de transações e gerar ou atualizar uma raiz *hash* pelo mecanismo *Patricia Merkle Tree*;
- **segunda validação:** invariabilidade do *hash* gerado pelo mecanismo *Patricia Merkle Tree* independentemente da ordem das transações organizadas no bloco;
- **terceira validação:** independência quanto à necessidade de recuperação da cadeia completa para consistir o *hash* do bloco quando se utiliza o mecanismo *Patricia Merkle Tree*;
- **quarta validação:** aprimoramento com relação à capacidade de armazenamento de dados com o mecanismo *Patricia Merkle Tree* por organizar o índice da árvore considerando repetições de dados nos objetos de sua estrutura.

Os resultados destas validações são apresentados e detalhados no item 4.4.1.

4.3.2 Cenário 2: Desempenho do Consenso Híbrido

O objetivo principal desta seção é estabelecer o cenário para demonstrar que o mecanismo de consenso híbrido proposto no modelo *Bankchain* tem melhor desempenho e resolve um dos principais impedimentos para utilização do *blockchain* em soluções de negócios, como no cenário das operações interbancárias, que é a questão do baixo desempenho e alta latência de resposta devido ao esforço do mecanismo de prova de trabalho.

Assim, foram confrontados os dois algoritmos, o *bankchain.py* que estabelece um *blockchain* com mecanismo de consenso híbrido seguindo as especificações da seção 3.2.1.2 desta dissertação, e o *brucecoin.py* que utiliza o mecanismo de prova de trabalho similar ao utilizado pelo *Bitcoin* para consenso dos nós com relação à cadeia, conforme estabelecido no item 4.2.2.

Ambos os protótipos possuem os seguintes microsserviços implementados:

- ***get chain*:** chamada *REST* pelo método *GET* para recuperar a cadeia do *blockchain*;
- ***connect node*:** chamada *REST* pelo método *POST* para realizar a conexão dos nós, conforme *JSON* estabelecido no *body* com a lista dos nós ativos na rede;
- ***add transaction*:** chamada *REST* pelo método *POST* para adicionar transações ao próximo bloco a ser minerado, conforme *JSON* escrito no *body* da requisição com os dados da transação;
- ***mine block*:** chamada *REST* pelo método *GET* para minerar o bloco com as transações disponíveis;
- ***replace chain*:** chamada *REST* pelo método *GET* para atualizar a cadeia mantida pelo nó pela execução do mecanismo de consenso, que na implementação *Brucecoin* considera apenas o tamanho da cadeia e a validade dos blocos adicionados a ela, e na implementação *Bankchain* além destas considerações, acrescenta três rodadas de validação por chamadas a nós distintos;
- ***is valid*:** chamada *REST* pelo método *GET* para verificar se a cadeia é válida.

A principal diferença entre as implementações *bankchain.py* e *brucecoin.py* devem-se ao mecanismo de mineração e consenso adotados e às informações estabelecidas nos blocos da cadeia. Sendo que o *bankchain.py* utiliza seu mecanismo de consenso híbrido específico e os blocos possuem informações adequadas ao proposto para as comunicações de operações interbancárias.

Já o *brucecoin.py* executa a prova de trabalho no processo de mineração dos blocos e não inclui a verificação por outros nós no mecanismo de consenso, além de que os blocos possuem informações mais próximas ao contexto das criptomoedas, sendo assim mais similar à implementação *Bitcoin* [1], questão que não afeta os termos comparativos.

As validações serão apresentadas na seção 4.4.2.

4.4 VALIDAÇÕES DOS CENÁRIOS DE TESTES

Nesta seção são apresentados os resultados obtidos com a execução dos protótipos implementados para atender aos cenários estabelecidos no item 4.3.

4.4.1 Validações do Cenário 1: Escalabilidade com *Patricia Merkle Tree*

A primeira validação é da capacidade de reunir um grupo de transações e gerar ou atualizar uma raiz *hash* utilizando o protótipo do mecanismo *Patricia Merkle Tree*. Assim, a execução do *script* conforme a imagem 4.1 demonstra a geração da raiz *hash* a partir de um conjunto de transações apropriadas no formato *JSON* em um par índice e conteúdo para cada transação.

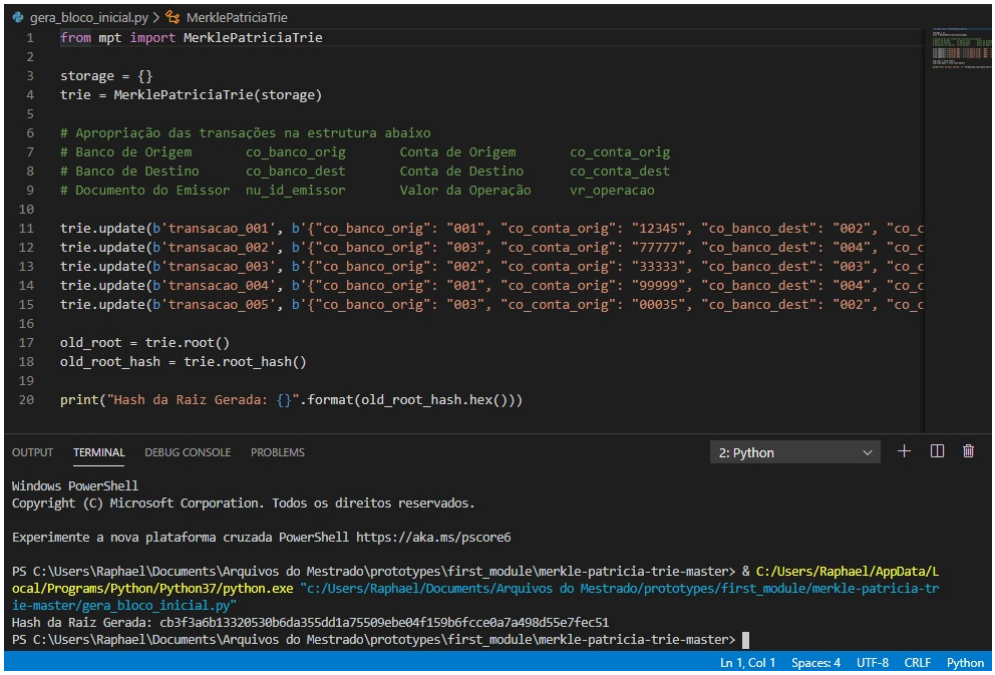

Figura 4.1: Execução do *script Python* para geração do *hash Patricia Merkle Tree*

Já a segunda validação é a de que com o mecanismo *Patricia Merkle Tree* independentemente da ordem das transações organizadas no bloco, o *hash* gerado é sempre o mesmo. Assim, neste caso, já é possível estabelecer uma vantagem em relação ao *Merkle Tree* utilizado no *Bitcoin*, modelo de referência da tecnologia *blockchain*, visto que esse apresenta diferença nos *hashes* quando a quantidade de transações gera um número ímpar de nós folhas.

O *hash* raiz de uma árvore de *Merkle* é o último de um encadeamento de pares, em que cada um fornece o *hash* que é operado ao *hash* de outro par até que os últimos dois pares formem o *hash* raiz. Desta forma, quando as transações geram um número ímpar de nós folhas, o ultimo nó folha é duplicado para manter a coerência da estrutura de dados. Porém, o efeito colateral é que neste caso a ordem dos nós folhas gera diferença no *hash* raiz, assim *hashes* diferentes são gerados para os mesmo objetos à depender da sequência em que eles estão organizados. Esse é um problema do *Bitcoin* [59] por utilizar o *Merkle Tree* que pode ser superado com a substituição pelo *Patricia Merkle Tree*. Assim, as imagens 4.2 e 4.3 mostram

os resultados dos testes realizados para comprovação que na implementação do protótipo a sequência das transações pares não impacta.

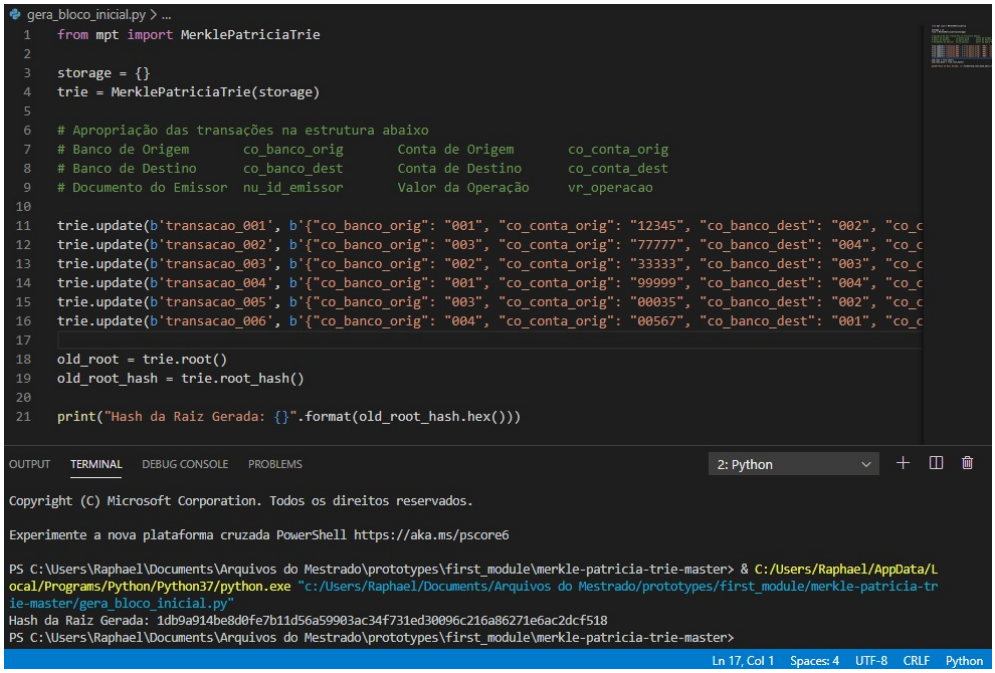

Figura 4.2: Execução do *script* com as transações ordenadas

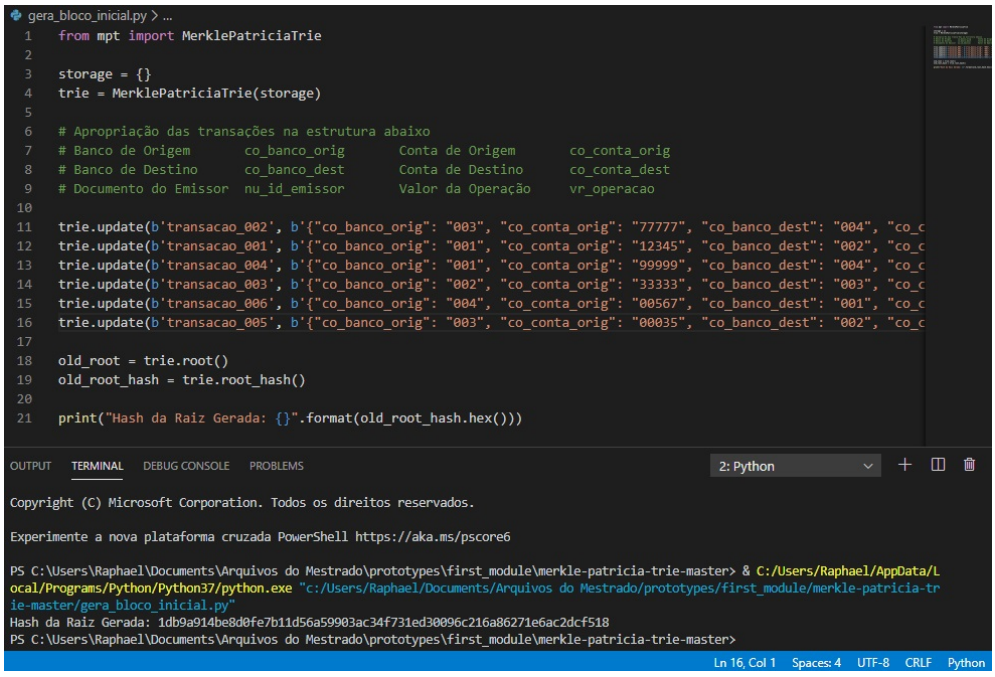

Figura 4.3: Execução do *script* com as transações desordenadas (em ordem distinta)

A terceira validação obtida pela implementação do *Patricia Merkle Tree* é a de independência quanto à necessidade de recuperação da cadeia completa para consistir o *hash* do bloco. Diferentemente do *Bitcoin*, por exemplo, que exige o *download* completo da cadeia para consistência do *hash*, gerando assim um problema de escalabilidade, pois todos os nós participantes terão que baixar uma cadeia cada vez maior

para participar da rede, conforme abordado no item 2.5 e citado nos artigos [60], [61] e [62].

Assim, as imagens 4.4 e 4.5 demonstram na implementação do *Patricia Merkle Tree* que tendo o *hash* do bloco, a raiz é atualizada independentemente da entrada ser a cadeia completa ou não.

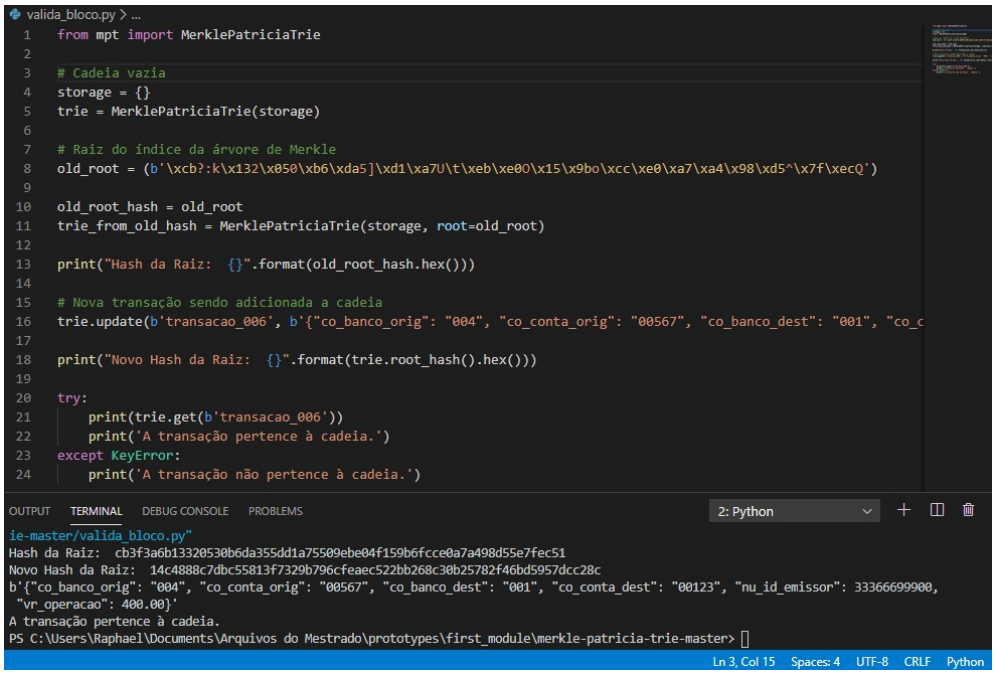

Figura 4.4: Execução do *script* com a cadeia vazia

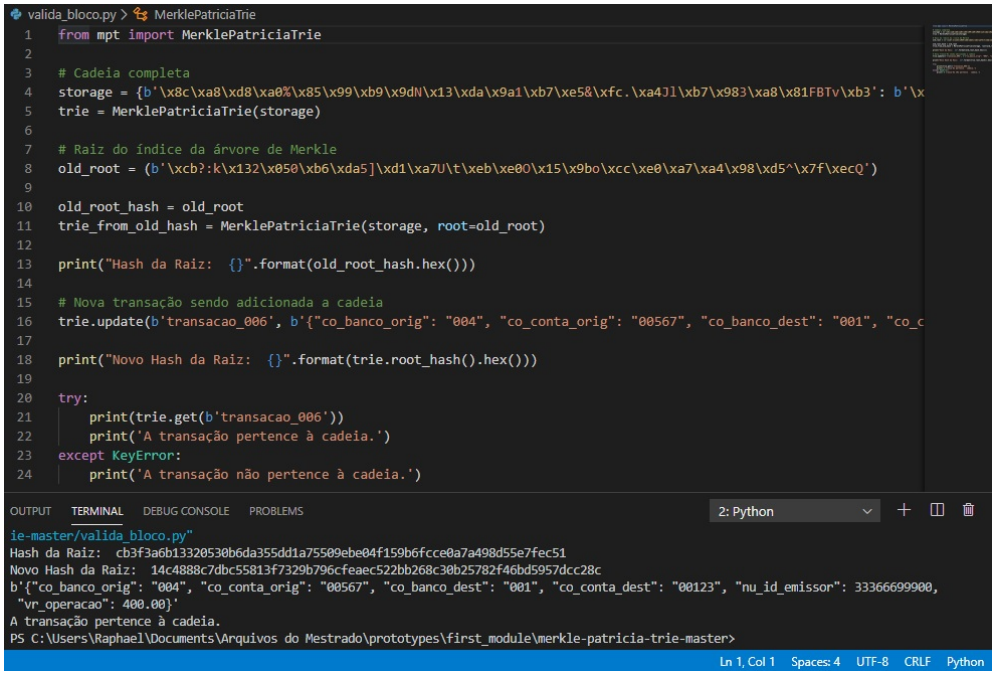

Figura 4.5: Execução do *script* com a cadeia completa

Desta forma, a primeira imagem 4.4 demonstra a geração do *hash* de confirmação com a cadeia vazia, enquanto a segunda imagem 4.5 executa a geração do *hash* com a cadeia completa, em ambos os casos a partir da cadeia gerada conforme a figura 4.1. Comprovando, assim, que com o *Patricia Merkle Tree*

não há necessidade de *download* da cadeia toda para fazer mineração na rede, basta o índice atualizado da árvore de *Merkle* ser compartilhado com o nó participante para que este consiga atualizar a cadeia e gerar seu novo *hash*.

E a quarta validação apresentada nesse item sobre as otimizações obtidas com a substituição do *Merkle Tree* pelo *Patricia Merkle Tree* com relação ao *hash* de encadeamento dos blocos é referente a maior eficiência no armazenamento de dados. Isto se dá porque pela maneira como o mecanismo *Patricia Merkle Tree* é estruturado, conforme abordado no item 2.7.2, há uma otimização com relação ao armazenamento de dados por organizar o índice da árvore considerando que repetições de dados nos objetos da estrutura.

Desta forma, os quatro cenários estabelecidos no item 4.3.1 para validar a escalabilidade obtida com a utilização do mecanismo *Patricia Merkle Tree* proposto para o modelo *Bankchain* geraram resultados com o objetivo respectivamente de: a) verificar o tamanho do índice da árvore considerando duas transações, b) obter o tamanho do índice para quatro transações, c) atualizar a cadeia com as mesma duas primeiras transações duplicadas, chave e conteúdo, e analisar o tamanho do índice, e d) analisar a cadeia com as transações duplicando apenas conteúdo.

Assim, a imagem 4.6 contempla o resultado do primeiro cenário, a figura 4.7 trata do segundo, o *print* 4.8 é referente ao terceiro cenário, e a imagem 4.9, ao quarto. Pela análise dos resultados, considerando a informação do tamanho do índice no console do terminal Python na execução dos *scripts*, conforme as figuras 4.6, 4.7, 4.8 e 4.9, o tamanho do índice aumenta proporcionalmente ao número de transações, porém é otimizado no caso em que as informações são repetidas na árvore de dados, tanto que o tamanho dobrou quando as transações dobraram, permaneceu o mesmo quando foram completamente duplicadas, e manteve um tamanho intermediário quando o índice foi alterado e o conteúdo duplicado.

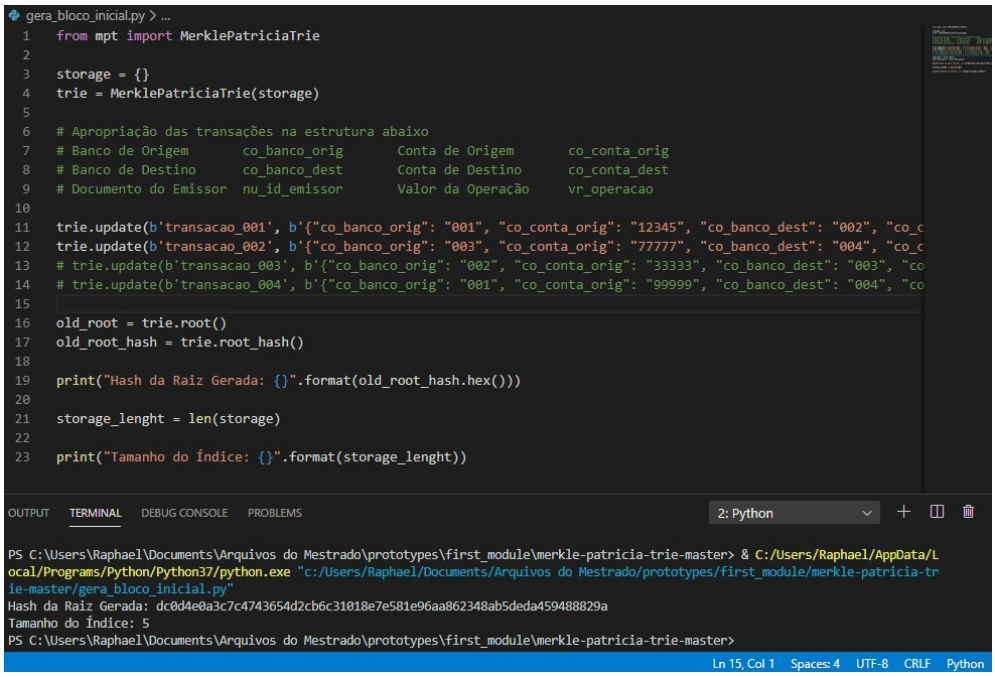

Figura 4.6: Cenário com duas transações

Em resumo, as validações estabelecidas pela utilização da implementação do mecanismo *Patricia Merkle Tree* embasam a premissa estabelecida no modelo *Bankchain* ao adotá-lo em sua arquitetura de que há

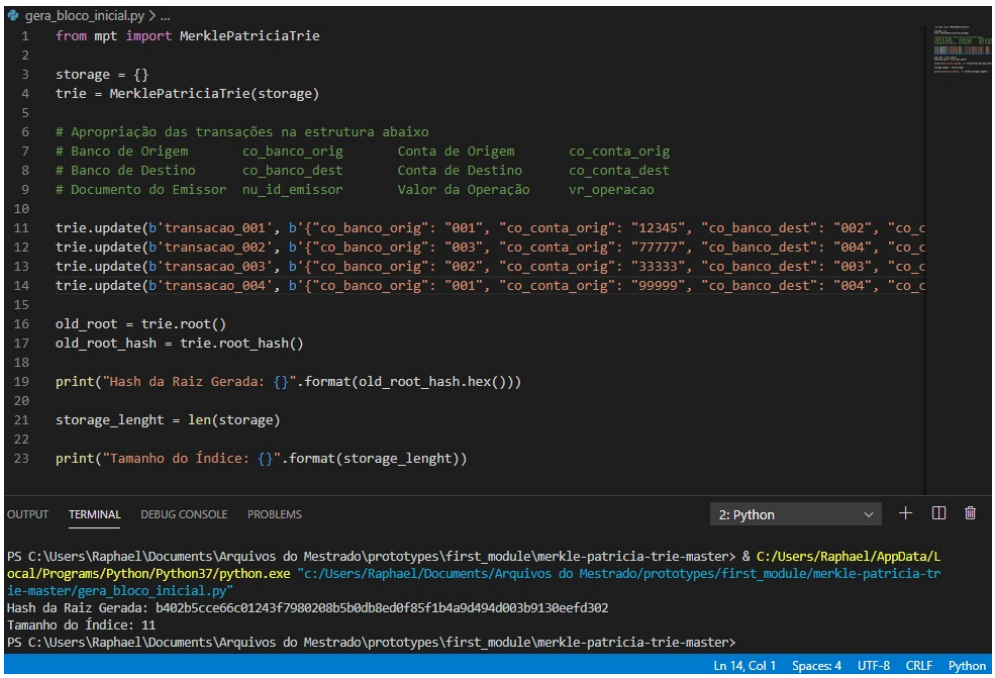

Figura 4.7: Cenário com quatro transações

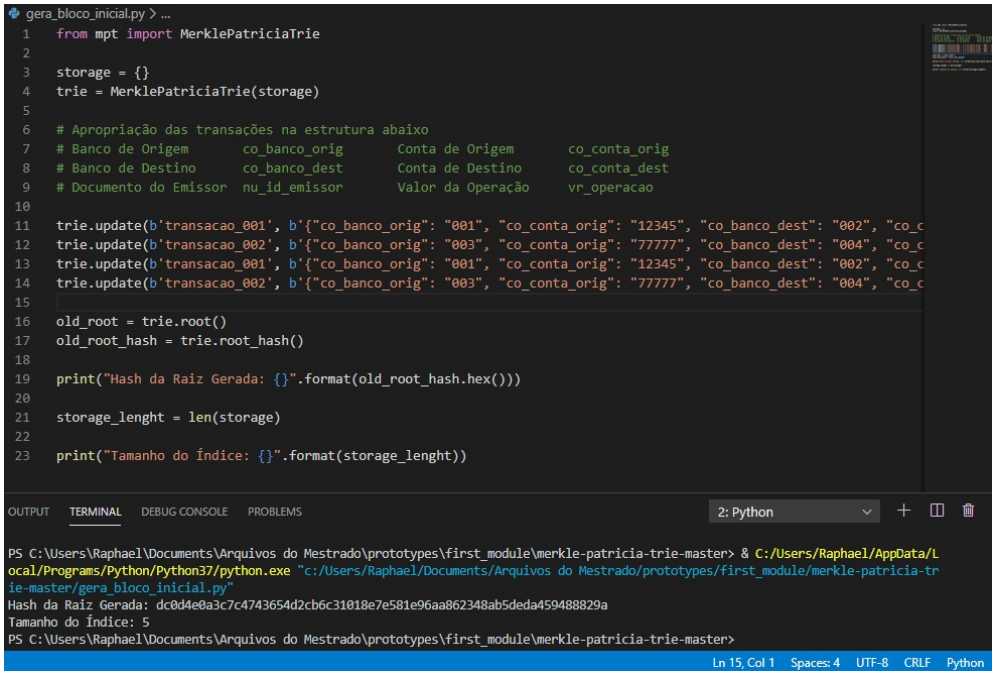

Figura 4.8: Cenário com duas transações duplicadas, índice e conteúdo

ganho de escalabilidade em comparação ao modelo mais usual de *blockchain*, estabelecido pela implementação conforme o *Bitcoin* utilizando o *Merkle Tree* tradicional. Esta otimização de escalabilidade pode ser considerada pela não necessidade de recuperação da cadeia completa, permitindo que novos nós sejam incorporados à rede sem a latência do *download* da cadeia completa, e pela economia de espaço com relação ao armazenamento dos dados, pois, diferentemente do mecanismo utilizado no *Bitcoin*, neste a estrutura da árvore considera o conteúdo para armazenamento, que ocuparam a mesma *branch* caso sejam iguais.

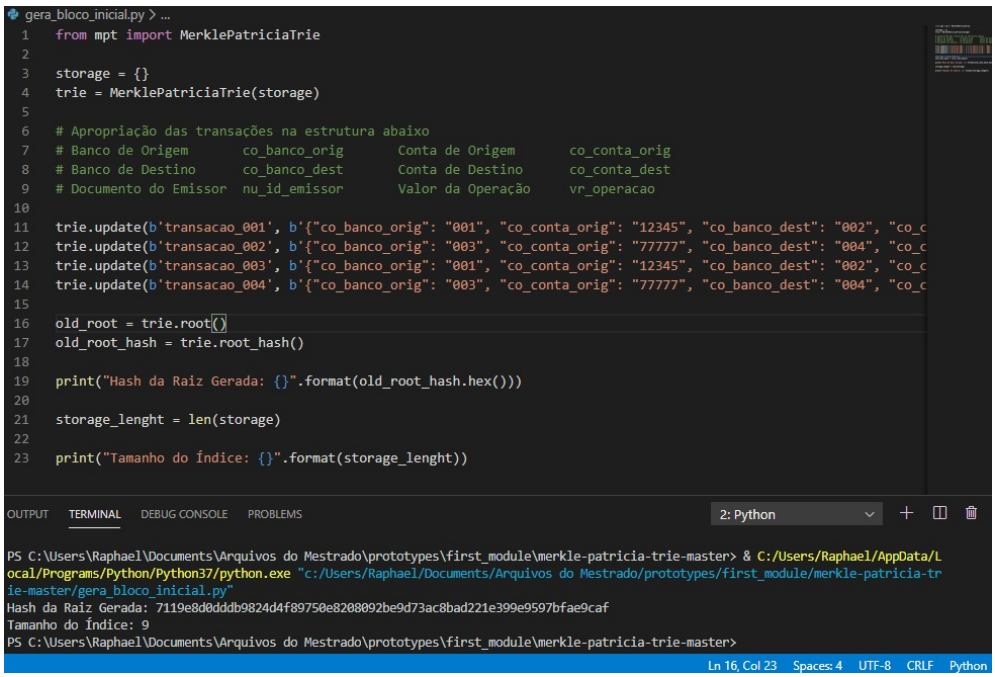

Figura 4.9: Cenário com duas transações duplicadas, apenas conteúdo

Além do ganho de escalabilidade, o mecanismo do *Patricia Merkle Tree* ainda possui a vantagem de superar o problema dos nós folha em quantidade ímpar, permitindo a comparação do *hash* das transações independentemente da ordem de apropriação.

4.4.2 Validações do Cenário 2: Desempenho do Consenso Híbrido

Neste item são exploradas as validações obtidas ao se confrontar as implementações *bankchain.py* e *brucecoin.py*, esta obedecendo aos paradigmas de consenso do *Bitcoin* [1] e aquela às propostas do mecanismo híbrido da arquitetura de solução proposta nesta dissertação, conforme item 4.2.2.

Uma questão importante no contexto dessa validação de desempenho é a referência com relação ao algoritmo da prova de trabalho, sendo que a substituição deste por uma solução pertinente ao contexto de um *blockchain* de consórcio para comunicações interbancárias foi fundamental. Tanto que mesmo em um contexto, em que a prova de trabalho seja bem mais simples que, por exemplo, a exigida atualmente no *Bitcoin*, o *Bankchain* apresentou resultados mais robustos e consistentes.

Como a prova de trabalho estabelecida para o *Bitcoin* consiste atualmente na geração de blocos cujos *hash* das transações iniciem com 19 zeros à esquerda, algo fora de questão para uma atividade de verificação utilizando o ambiente disponível, conforme estabelecido na seção 4.2, pois a mineração de um bloco utilizando este paradigma na prova de trabalho levaria dias executando a mineração de um bloco apenas, foi definida uma prova de trabalho de menor custo para a implementação.

Assim, avaliou-se o comportamento do algoritmo de prova de trabalho em cenários de 4 à 8 zeros à esquerda no *hash* das transações do bloco, conforme resultados da tabela 4.3, e optou-se por uma implementação que exigisse 5 zeros à esquerda como prova de trabalho para viabilizar a execução comparativa entre as implementação.

Tabela 4.3: Resultados da mineração de blocos com 1 transação cada por 3 nós distintos

Quantidade de Zeros	Tempo Médio de Execução
4 Zeros	13,33 ms
5 Zeros	2,36 s
6 Zeros	12,46 s
7 Zeros	12 m 36 s
8 Zeros	1 h 54 m

A título de ilustração, o bloco gênesis do *Bitcoin* possui 10 zeros à esquerda e no ambiente de testes disponível demorou 86 horas para minerar um bloco com esta exigência.

Analisando o gráfico 4.10 é perceptível o comportamento mais consistente do *Bankchain*, que se mostrou otimizado ao tamanho da rede, algo interessante para o contexto de consórcio, que conta com um número limitado de participante e possui crescimento controlado.

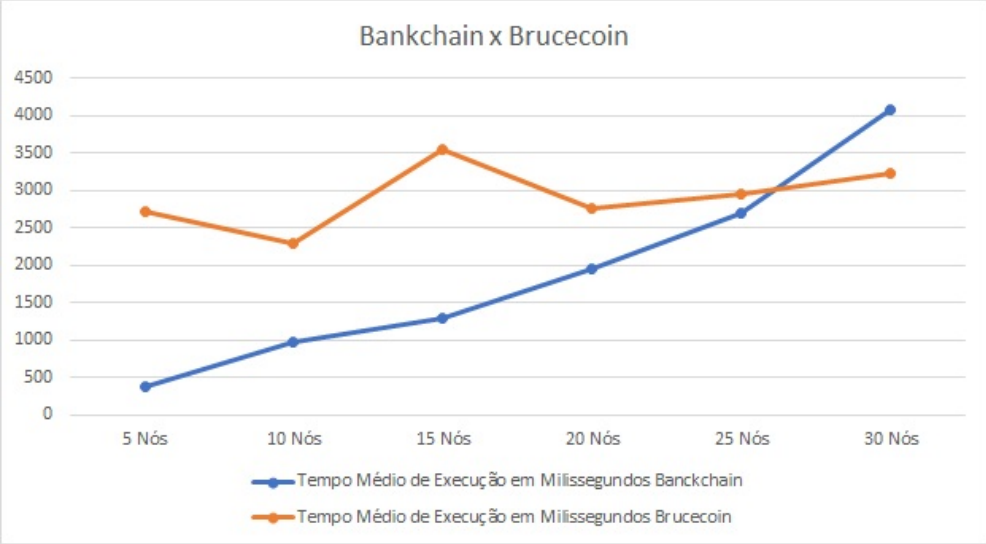

Figura 4.10: Gráfico comparativo dos tempos médios de execução das requisições de cada rodada

Assim, o desempenho no *Bankchain* variou proporcionalmente com o aumento dos nós na rede, além de estabelecer uma solução cujo tempo de resposta foi melhor nos cenários em que é mais adequada, mesmo em um contexto de teste que a prova de trabalho exigida no *Brucecoin* é bem simples e a conferência dos resultados por mais da metade da rede não foi exigida.

A verificação de desempenho consistiu na execução dos métodos ***get chain***, ***connect node***, ***add transaction***, ***mine block*** e ***replace chain*** expostos utilizando o *Flask* por cada uma das implementações, através de chamadas *REST* pelo *Postman*. Foram delimitadas seis rodadas de testes nas quais novos nós eram incluídos na rede e o ciclo das requisições reiniciado, em todas eram executadas as seguintes ações através dos respectivos microsserviços expostos nesta sequência:

- 1º: conexão dos nós (***connect node***);
- 2º: recuperação da cadeia (***get chain***);
- 3º: inclusão de 2 transações (***add transaction***);

- 4º: mineração do bloco (***mine block***);
- 5º: consenso sobre a cadeia da rede (***replace chain***).

Como exemplo da forma de execução e mensuração dos tempos de resposta, nas imagens 4.11 e 4.12 temos os *prints* da execução do método ***mine block*** no *node* 5001 na última rodada e do método ***replace chain*** no *node* 5030 também na última rodada, respectivamente, na implementação *Bankchain*.

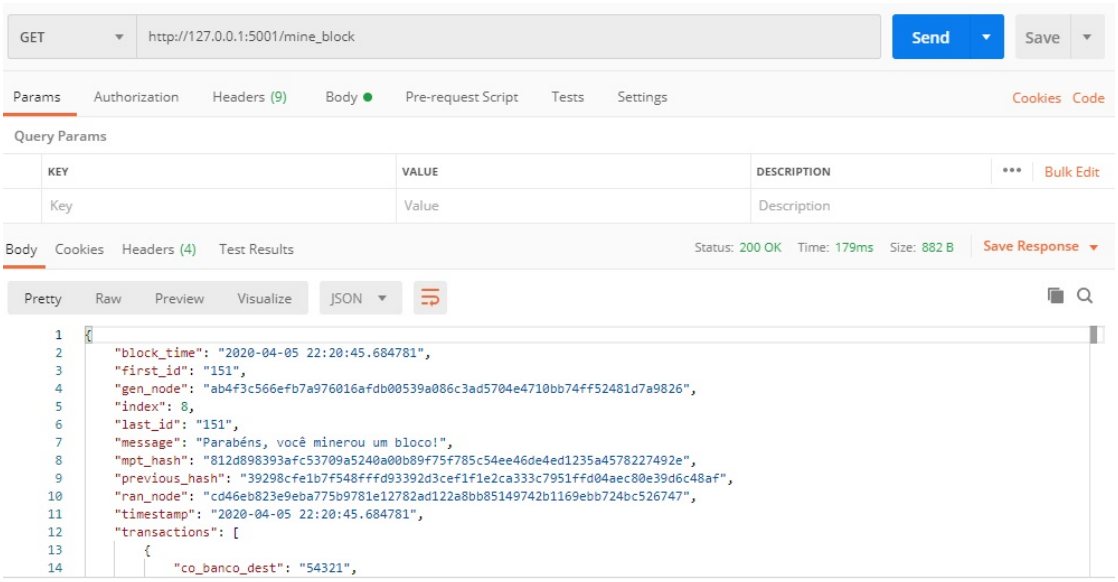

Figura 4.11: Execução do método ***mine block*** no *node* 5001

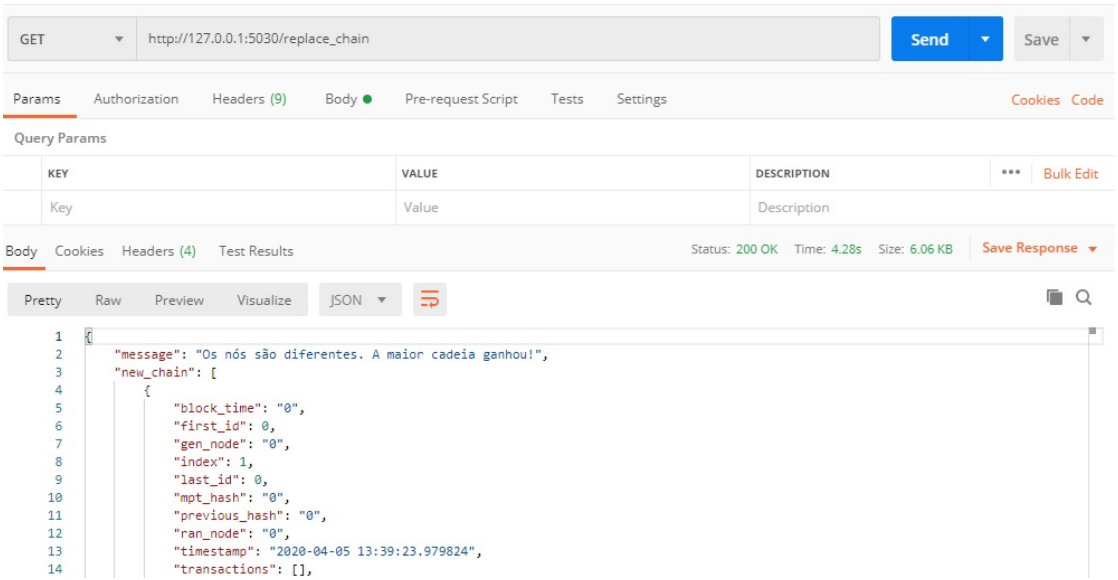

Figura 4.12: Execução do método ***replace chain*** no *node* 5030

As rodadas tinham 5, 10, 15, 20, 25 e 30 nós, respectivamente, sendo coletado o tempo de execução de cada uma das requisições de cada nó. Para avaliação dos resultados foi calculada a média do tempo de execução das requisições para obter o tempo médio total da rodada em cada implementação.

Assim, a tabela 4.4 planilha o tempo médio de execução em milissegundos das rodadas, conforme a quantidade de nós, em cada uma das implementações. Algo evidenciado anteriormente de maneira comparativa no gráfico 4.10, em que é possível visualizar os resultados consolidados.

Tabela 4.4: Tempo médio de execução em milissegundos do total das requisições em cada rodada

Quantidade de Nós	*Bankchain*	*Brucecoin*
5	1381,1	2712,7
10	969,55	2297,75
15	1290,9	3545,86
20	1961,12	2752,1
25	2690,52	2962,04
30	4074,48	3240,13

Já a tabela 4.5 apresenta o tempo médio de execução em milissegundos de cada função nos testes realizados para cada implementação.

Tabela 4.5: Tempo médio de execução em milissegundos de cada função

Função Executada	*Bankchain*	*Brucecoin*
get chain	30,75	44,48
connect node	35,66	71,325
add transaction	35,63	39,35
mine block	38,7	2162,11
replace chain	1757,48	601,15

E os gráficos 4.13 mostram os resultados da tabela 4.5.

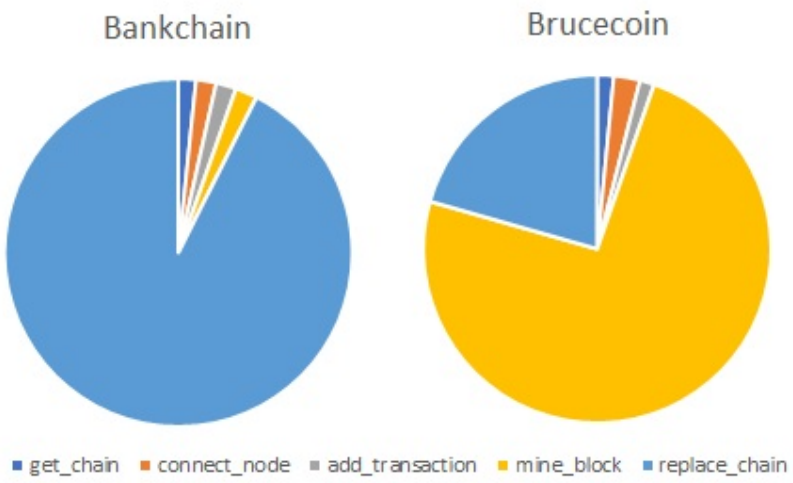

Figura 4.13: Gráfico comparativo dos tempos médios de execução das funções de cada implementação

Os resultados comprovam que a implementação conforme o modelo proposto nesta dissertação com a utilização do mecanismo de consenso híbrido estabelecido para utilização em *blockchain* de consórcio apresenta um crescimento proporcional à quantidade de nós, diferentemente da implementação mais próxima ao modelo *Bitcoin*, que apresentou tempos de execução mais erráticos com o aumento da quantidade de nós participantes, conforme evidencia o gráfico 4.10.

Além de que os tempos de respostas das funções em cenários com poucos nós, como acontece no contexto das *blockchains* de consórcios, o modelo proposto *Bankchain* obteve resultados bem melhores comparativamente. Algo que é bastante positivo considerando que a proposta é para um modelo de governança de consórcios em que a quantidade de nós é reduzida, visto que os participantes serão apenas os parceiros que mantêm a rede para efetivação de comunicações das operações interbancárias relizadas entre estas entidades.

Importante observar também que as funções impactadas foram diferentes, no *Bankchain* o método ***replace chain*** foi o mais oneroso, justamente por exigir a verificação por nós específicos no mecanismos de consenso. Já na implementação *Brucecoin* a checagem foi apenas pelo tamanho da cadeia, não exigindo verificação da maioria da rede, como o *Bitcoin*, o que oneraria mais o método, mas que não foi estabelecido no contexto dessa implementação para que ambos os escopos fossem equalizados para utilização mais próxima de *blockchain* de consórcio.

E na implementação *Brucecoin* o principal impacto foi no método ***mine block***, que apesar de exigir uma prova de trabalho bem menos complexa que a do *Bitcoin*, ainda apresentou tempo elevado de resposta e alto consumo de *CPU*, problemas clássicos da implementação de referência, conforme apresentado na discussão do item 2.5, e não encontrados na implementação *Bankchain*.

Assim, os resultados validam a implementação *Bankchain* como mais consistente visto que aumenta o tempo de resposta proporcionalmente ao acréscimo da quantidade de nós, uma vantagem ao possibilitar um planejamento mais otimizado conforme a quantidade de participantes na rede. Além de ter melhor desempenho que a implementação *Brucecoin*, principalmente no contexto mais próximo à utilização por *blockchain* de consórcio. Mesmo que para os testes a dificuldade da prova de trabalho exigida no *Brucecoin* foi bem menos complexa que o do *Bitcoin* e não foi exigida a verificação pela maioria da rede, outra exigência da famosa implementação [1].

4.5 VALIDAÇÕES

Nesta seção dissertarei sobre os resultados apresentados em 4.4 com a execução dos protótipos implementados para atender aos cenários estabelecidos em 4.3, considerando as motivações e os objetivos da arquitetura de solução proposta nesta dissertação no item 3.2.

A proposta explorada nessa dissertação atende ao objetivo de estabelecer um modelo de uso de *blockchain* para comunicação de operações interbancárias entre entidades financeiras com alta disponibilidade, baixa latência, auditoria constante e independência de órgão terceiro por apresentar uma arquitetura de solução que resolve problema conhecidos para principais implementações *blockchain*, principalmente com relação aos gargalos de escalabilidade e desempenho.

Assim, o modelo proposto traz como principais contribuições a substituição do padrão *Merkle Tree* pelo *Patricia Merkle Tree* como algoritmo de estruturação de dados da camada de consenso para garantir melhor escalabilidade pela eficiência de armazenamento de dados, além da substituição dos padrões de mecanismo de prova de trabalho e de prova de risco por um mecanismo híbrido que utiliza conceitos da prova de autoridade e do *Practical Byzantine Fault Tolerance* com o intuito de superar obstáculos de

desempenho conhecidos do padrão *Bitcoin*.

Os protótipos desenvolvidos e os cenários executados possuem justamente os objetivos de atender estes enfoques e os resultados apresentados demonstram que o modelo proposto para utilização da *blockchain* na comunicação de operações interbancárias entre entidades financeiras atende às expectativas e possui enorme potencial neste contexto, pois o *Patricia Merkle Tree* respondeu conforme os comportamentos esperados para a proposta superando limitações conhecidas do *Merkle Tree*, principalmente a necessidade de *download* da cadeia completa de transações para obter o *hash* da raiz. Assim como o mecanismo de consenso híbrido do modelo teve melhor desempenho quando comparado aos algoritmos do padrão *Bitcoin*, respeitando os paradigmas de *blockchain* e tendo desempenho superior principalmente no contexto com poucos nós como o das operações interbancárias.

5 CONCLUSÃO

Nesta dissertação, a tecnologia *blockchain* foi pesquisada e debatida frente à perspectiva de solução para a questão de confiança nas interoperações entre diferentes entidades. Assim, justificada pelo potencial desta tecnologia e embasada pelos estudos de seus mecanismos, conforme apresentado no capítulo 2, foi estabelecida uma arquitetura para utilização da tecnologia *blockchain* na comunicação das operações interbancárias. Utilizando esta em um cenário prático de atuação negocial para compartilhar dados de forma confiável, em que fossem superadas questões específicas de escalabilidade, desempenho e privacidade.

O *Blockchain* como um banco de dados distribuído cujos registros são organizados em blocos encadeados compartilhados em uma rede *peer-to-peer* é um conceito novo, mas já bastante conhecido. E diversas são as possibilidades ventiladas de aplicações e várias as discussões sobre o precursor *Bitcoin*, porém pouco se conhece sobre os mecanismos intrínsecos da tecnologia e as possibilidades de adaptação conforme o modelo de governança adotado e a necessidade de uso específica.

Assim, esta dissertação buscou atuar justamente nessa questão, explorando as características da tecnologia e ponderando sobre suas vantagens e desvantagens para propor uma arquitetura que atendesse um modelo de governança determinado, atuasse frente a um problema negocial específico e adotasse os mecanismos coerentes para a solução o qual se propõe.

Desta forma, no capítulo 3 foi exposta a necessidade de comunicação das operações interbancárias entre as entidades financeiras e como as soluções tradicionais adotam a troca de arquivos em lote por rotinas *batch* como mecanismo, sem estabelecer uma visão compartilhada e integrada do fluxo de negócio, além de estar frequentemente sujeita à problemas de disponibilidade e falta de transparência.

Considerando esta situação e a necessidade de utilização da tecnologia *Blockchain* em soluções de negócios, além do contexto das criptomoedas, foi estabelecida a arquitetura de solução proposta, denominada *Bankchain*, utilizando *blockchai*n para prover uma rede distribuída *peer-to-peer* para comunicação de operações interbancárias entre diferentes entidades com garantia de escalabilidade, desempenho e privacidade.

Para atender à estas pretensões, o *Bankchain* foi estabelecido para uso na governança de consórcio, adotando o *Patricia Merkle Tree* em substituição ao *Merkle Tree* tradicional no *hash* de encadeamento dos blocos, a prova de autoridade para permissionamento das ações e o *PBFT* para validação da cadeia formulando assim um mecanismo de consenso híbrido, além de diretivas de mercado para privacidade dos dados e os padrões de *smart contract* para auditoria da rede.

Entendendo que as propostas de uso do *Patricia Merkle Tree* e o mecanismo de consenso híbrido eram os principais diferenciais do modelo, que trariam os ganhos de escalabilidade e desempenho que habilitariam a utilização da *Blockchain* em um problema de negócio prático como a comunicação das operações interbancárias entre instituições financeiras, assim foram delimitados protótipos do modelo à serem desenvolvidos e testados para comprovação dos benefícios da solução.

Consequentemente, no capítulo 4 são expostas as considerações quanto às implementações realizadas para comprovação de escalabilidade e desempenho do modelo *Bankchain*, comparando-as com relação aos

mecanismos adotados pela solução mais tradicional da tecnologia *blockchain*, a qual é referência para a maioria das plataformas e soluções implementadas, o *Bitcoin* [1].

Assim, três protótipos foram desenvolvidos: o do algoritmo de manutenção do *hash* utilizando o *Patricia Merkle Tree*, a *blockchain Banckchain* conforme as propostas de estrutura de encadeamento e dados, validação e consenso do capítulo 3 da dissertação, e a *blockchain Brucecoin* implementação utilizando a prova de trabalho e o consenso pela comparação do tamanho da rede de forma similar ao *Bitcoin*, porém sem exigir a verificação da maioria da rede como este famoso modelo.

Dois cenários distintos de comprovação foram estabelecidos e os resultados obtidos foram bastante satisfatórios. O primeiro cenário foi o da comprovação de que o *Patricia Merkle Tree* proporciona melhor escalabilidade à *blockchain* por não precisar que todos os nós tenham a cadeia completa para conseguir gerar o próximo *hash*, bastando ter a informação do último, e de que sua estruturada de dados é mais eficiência quanto ao armazenamento. E o segundo cenário foi o da comprovação de que o mecanismo híbrido de consenso proposto é mais consistente e de melhor desempenho que a implementação *Brucecoin*, principalmente no contexto mais próximo à utilização por *blockchains* de consórcio com até 25 nós, por não ter a necessidade de execução da prova de trabalho.

Concluindo, esta dissertação contribui como um estudo dos mecanismos *blockchain* e quanto a possibilidade de adotar as melhores opções para estabelecer uma arquitetura de solução adequada a um problema específico, o que ajuda na ampliação do uso da tecnologia como solução negocial prática. Além de contribuir com a proposição de uma abordagem mais eficiente de encadeamento dos dados em *blockchain* com a utilização do *Patricia Merkle Tree* e com a formulação de um mecanismo híbrido de consenso adequado às soluções de governança de consórcio.

5.1 TRABALHOS FUTUROS

Conforme exposto nessa conclusão e a abordagem definida no capítulo 4, o modelo proposto como solução no capítulo 3 foi além das abordagem implementadas nos protótipos, principalmente devido às limitações do ambiente de laboratório e ao uso de padrões e diretrizes consolidadas, que não acrescentariam aos objetivos do trabalho.

Porém, é preciso avançar na análise com relação aos aspectos de privacidade de dados e auditoria da rede e como as soluções adotadas nestas perspectivas podem influenciar na escalabilidade e no desempenho do modelo *Bankchain*. Assim, são sugeridos, como proposta de trabalhos futuros, os itens:

- Analisar de forma mais profunda os problemas de privacidade de dados em soluções *blockchain*;
- Avaliar a execução de *smart contracts* fora das plataformas proprietárias;
- Aprimorar o modelo proposto com a implementação das funcionalidade de privacidade de dados e auditoria da rede através do módulo de gerenciamento do cadastro e das ações dos participantes da rede;
- Estudar utilização de inteligência artificial para manutenção dos parâmetros de privacidade de dados

e auditoria da rede conforme os dados mantidos pelo módulo de gerenciamento do cadastro e das ações dos participantes da rede.

A exploração de trabalhos futuros significa descobrir necessidades de soluções ao modelo de arquitetura proposto não contempladas nesta dissertação. *Blockchain* é um campo fértil para inovações e seria ilusório assumir que apenas um trabalho de pesquisa possa cobrir todo o assunto. Nesse sentido, esta dissertação pode ser considerada uma contribuição no contexto de um campo de pesquisa em aberto e possivelmente uma colaboração para pesquisa e inovação sobre o tema.

REFERÊNCIAS BIBLIOGRÁFICAS

1 NAKAMOTO, S. et al. Bitcoin: A peer-to-peer electronic cash system. Working Paper, 2008.

2 ROCCHINI, C. Patricia trie (radix tree) sample: latinus word. Disponível em: <https://en.wikipedia.org/wiki/Radix_tree>.

3 LEOW, A. Common mistakes to avoid in enterprise blockchain projects. *Gartner Report*, Gartner, 05 2019.

4 TAMA, B. A.; KWEKA, B. J.; PARK, Y.; RHEE, K.-H. A critical review of blockchain and its current applications. In: IEEE. *2017 International Conference on Electrical Engineering and Computer Science (ICECOS)*. [S.l.], 2017. p. 109–113.

5 ZHENG, Z.; XIE, S.; DAI, H.; CHEN, X.; WANG, H. An overview of blockchain technology: Architecture, consensus, and future trends. In: IEEE. *2017 IEEE International Congress on Big Data (BigData Congress)*. [S.l.], 2017. p. 557–564.

6 KAUSHIK, A.; CHOUDHARY, A.; EKTARE, C.; THOMAS, D.; AKRAM, S. Blockchain—literature survey. In: IEEE. *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. [S.l.], 2017. p. 2145–2148.

7 BELOTTI, M.; BOŽIĆ, N.; PUJOLLE, G.; SECCI, S. A vademecum on blockchain technologies: When, which, and how. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 4, p. 3796–3838, 2019.

8 WÜST, K.; GERVAIS, A. Do you need a blockchain? In: IEEE. *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. [S.l.], 2018. p. 45–54.

9 MECHKAROSKA, D.; DIMITROVA, V.; POPOVSKA-MITROVIKJ, A. Analysis of the possibilities for improvement of blockchain technology. In: IEEE. *2018 26th Telecommunications Forum (TELFOR)*. [S.l.], 2018. p. 1–4.

10 LEOW, A.; LITAN, A.; HUNTER, R.; KANDASWAMY, R.; ARON, D.; STEVENS, A. Common mistakes to avoid in enterprise blockchain projects. *Gartner Report*, Gartner, 12 2018.

11 BRUCE, R.; JR., R. S.; MENDONçA, F. de; PIMENTEL, J.; HOLANDA, M.; FILHO, F. C. Blockchain para operações interbancárias. IADIS, p. 265–268, 2019.

12 TASCA, P.; THANABALASINGHAM, T.; TESSONE, C. J. Ontology of blockchain technologies. principles of identification and classification. *SSRN Electronic Journal*, v. 10, 2017.

13 LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, ACM, v. 4, n. 3, p. 382–401, 1982.

14 GRAMOLI, V. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, Elsevier, 2017.

15 ABRAHAM, I.; MALKHI, D. et al. The blockchain consensus layer and bft. *Bulletin of EATCS*, v. 3, n. 123, 2017.

16 YLI-HUUMO, J.; KO, D.; CHOI, S.; PARK, S.; SMOLANDER, K. Where is current research on blockchain technology?—a systematic review. *PloS one*, Public Library of Science, v. 11, n. 10, p. e0163477, 2016.

17 MATTILA, J. *The blockchain phenomenon–the disruptive potential of distributed consensus architectures*. [S.l.], 2016.

18 NGUYEN, G.-T.; KIM, K. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, v. 14, n. 1, 2018.

19 MINGXIAO, D.; XIAOFENG, M.; ZHE, Z.; XIANGWEI, W.; QIJUN, C. A review on consensus algorithm of blockchain. In: IEEE. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2017. p. 2567–2572.

20 WOOD, G. et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, v. 151, n. 2014, p. 1–32, 2014.

21 CLARKE, S.; CRAIG, I.; WYSZYNSKI, M. Litecoin cash: The best of all worlds sha256 cryptocurrency. *URL: https://litecoinca. sh/downloads/lcc_ whitepaper. pdf.*

22 EYAL, I.; GENCER, A. E.; SIRER, E. G.; RENESSE, R. V. Bitcoin-ng: A scalable blockchain protocol. In: *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*. [S.l.: s.n.], 2016. p. 45–59.

23 KING, S. Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, v. 1, p. 6, 2013.

24 KING, S.; NADAL, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, v. 19, 2012.

25 VASIN, P. Blackcoin's proof-of-stake protocol v2. *URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf*, v. 71, 2014.

26 COMMUNITY, N. Whitepaper:nxt. *URL: https://nxtwiki.org/wiki/Whitepaper:Nxt*.

27 TREW, C.; BRANDON, G.; DORIER, N. Stratis blockchain solutions. *URL: https://stratisplatform.com/.*

28 OLIVEIRA, M. T.; CARRARA, G. R.; FERNANDES, N. C.; ALBUQUERQUE, C. V.; CARRANO, R. C.; MEDEIROS, D. S.; MATTOS, D. M. Towards a performance evaluation of private blockchain frameworks using a realistic workload. In: IEEE. *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. [S.l.], 2019. p. 180–187.

29 ANGELIS, S. D.; ANIELLO, L.; BALDONI, R.; LOMBARDI, F.; MARGHERI, A.; SASSONE, V. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. 2018.

30 SZILáGYI, P. Clique poa protocol & rinkeby poa testnet. Disponível em: <https://github.com/ethereum/EIPs/issues/225>.

31 BUI, T.; COOPER, D.; COLLOMOSSE, J.; BELL, M.; GREEN, A.; SHERIDAN, J.; HIGGINS, J.; DAS, A.; KELLER, J.; THEREAUX, O. et al. Archangel: Tamper-proofing video archives using temporal content hashes on the blockchain. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2019. p. 0–0.

32 CACHIN, C. Architecture of the hyperledger blockchain fabric. In: *Workshop on distributed cryptocurrencies and consensus ledgers*. [S.l.: s.n.], 2016. v. 310, p. 4.

33 LIU, J.; LI, W.; KARAME, G. O.; ASOKAN, N. Scalable byzantine consensus via hardware-assisted secret sharing. *IEEE Transactions on Computers*, IEEE, v. 68, n. 1, p. 139–151, 2018.

34 CHEPURNOY, A.; DUONG, T.; FAN, L.; ZHOU, H.-S. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. *IACR Cryptology ePrint Archive*, v. 2017, p. 232, 2017.

35 MORRISON, D. R. Patricia—practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM (JACM)*, ACM, v. 15, n. 4, p. 514–534, 1968.

36 DAI, F.; SHI, Y.; MENG, N.; WEI, L.; YE, Z. From bitcoin to cybersecurity: A comparative study of blockchain application and security issues. In: IEEE. *2017 4th International Conference on Systems and Informatics (ICSAI)*. [S.l.], 2017. p. 975–979.

37 SASSON, E. B.; CHIESA, A.; GARMAN, C.; GREEN, M.; MIERS, I.; TROMER, E.; VIRZA, M. Zerocash: Decentralized anonymous payments from bitcoin. In: IEEE. *2014 IEEE Symposium on Security and Privacy*. [S.l.], 2014. p. 459–474.

38 BROWN, R. G.; CARLYLE, J.; GRIGG, I.; HEARN, M. Corda: an introduction. *R3 CEV, August*, v. 1, p. 15, 2016.

39 SALMAN, T.; ZOLANVARI, M.; ERBAD, A.; JAIN, R.; SAMAKA, M. Security services using blockchains: A state of the art survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 1, p. 858–880, 2018.

40 MAURER, F. K.; NEUDECKER, T.; FLORIAN, M. Anonymous coinjoin transactions with arbitrary values. In: IEEE. *2017 IEEE Trustcom/BigDataSE/ICESS*. [S.l.], 2017. p. 522–529.

41 ROCHA, S. A. Exchange of tax-related information and the protection of taxpayer rights: General comments and the brazilian perspective. *Bulletin for International Taxation*, v. 70, n. 9, p. 502–516, 2016.

42 ARMOUR, J.; AWREY, D.; DAVIES, P. L.; ENRIQUES, L.; GORDON, J. N.; MAYER, C. P.; PAYNE, J. *Principles of financial regulation*. [S.l.]: Oxford University Press, 2016.

43 FINCK, M. Blockchains: regulating the unknown. *German Law Journal*, Cambridge University Press, v. 19, n. 4, p. 665–692, 2018.

44 GRABOSKY, P. N. Regulation by reward: On the use of incentives as regulatory instruments. *Law & Policy*, Wiley Online Library, v. 17, n. 3, p. 257–282, 1995.

45 ALLEN, H. J. Regulatory sandboxes. *Geo. Wash. L. Rev.*, HeinOnline, v. 87, p. 579, 2019.

46 BANSAL, S. K.; BATRA, R.; JAIN, N. Blockchain the future of accounting. *The Management Accountant*, v. 53, n. 6, June 2018.

47 SZABO, N. Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)*, v. 18, p. 2, 1996.

48 YUAN, Y.; WANG, F.-Y. Blockchain and cryptocurrencies: Model, techniques, and applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, IEEE, v. 48, n. 9, p. 1421–1428, 2018.

49 ZHENG, Z.; XIE, S.; DAI, H.-N.; CHEN, X.; WANG, H. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, Inderscience Publishers (IEL), v. 14, n. 4, p. 352–375, 2018.

50 MIRANDA, J. C. de; ZUCHI, J. D. Tecnologia blockchain: a disrupção na indústria financeira. *Revista Interface Tecnológica*, v. 15, n. 2, p. 457–469, 2018.

51 Banco Central do Brasil. *Plataforma do BC com tecnologia blockchain facilitará troca de dados na supervisão do sistema financeiro*. 2018. Disponível em: <https://www.bcb.gov.br/detalhenoticia/249/noticia>. Acesso em: 02 janeiro 2022.

52 COSTA, P. M. da; SCHIMDT, B.; RODRIGUES, M. A. de M. Sistema financeiro digital. *Revista LIFT papers*, v. 1, n. 1, 2018.

53 YERMACK, D.; FINGERHUT, A. Blockchain technology's potential in the financial system. In: *Proceedings of the 2019 Financial Market's Conference*. [S.l.: s.n.], 2019.

54 Banco Central do Brasil. *PIX*. 2019. Disponível em: <https://www.bcb.gov.br/estabilidadefinanceira/pix>. Acesso em: 02 janeiro 2022.

55 Banco Central do Brasil. *Open Finance*. 2020. Disponível em: <https://www.bcb.gov.br/estabilidadefinanceira/openfinance>. Acesso em: 02 janeiro 2022.

56 INITIATIVE, M. O. S. Bitcoin core. Disponível em: <https://bitcoin.org/en/download>.

57 TURCIOS, R. A. S. t-student: Usos y abusos. *Revista mexicana de cardiología*, Asociación Nacional de Cardiólogos de México, Sociedad de Cardiología . . . , v. 26, n. 1, p. 59–61, 2015.

58 NEYMAN, J. X—outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, The Royal Society London, v. 236, n. 767, p. 333–380, 1937.

59 LAU, J.; MAXWELL, G.; DASHJR, L. Weaknesses in bitcoin's merkle root construction. *Merkle Report - Category: "Bitcoin"*, Merkle Report, 2019. Disponível em: <https://www.merklereport.com/category/bitcoin/page/77/>.

60 LITAN, A.; CLARK, W. The future of blockchain: 8 scalability hurdles to enterprise adoption. *Gartner Report*, Gartner, 9 2018.

61 MCCONAGHY, T.; MARQUES, R.; MÜLLER, A.; JONGHE, D. D.; MCCONAGHY, T.; MCMULLEN, G.; HENDERSON, R.; BELLEMARE, S.; GRANZOTTO, A. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB*, 2016.

62 ZHANG, K.; JACOBSEN, H.-A. Towards dependable, scalable, and pervasive distributed ledgers with blockchains. In: *ICDCS*. [S.l.: s.n.], 2018. p. 1337–1346.

bankchain.py

```
# -*- coding: utf-8 -*-

# Importing the libraries
import datetime
import hashlib
import json
import random
from flask import Flask, jsonify, request

from mpt import MerklePatriciaTrie

import requests
from uuid import uuid4
from urllib.parse import urlparse

class Blockchain:

    def __init__(self):
        self.chain = []
        self.transactions = []
        self.transactions_id = []

        initialize_arg = 0
        self.transactions_id.append(initialize_arg)

        self.chain_index = []
        self.chain_index.append(initialize_arg)

        self.validation_time = []
        self.validation_time.append(initialize_arg)

        self.create_block(previous_hash = '0', block_time = '0', gen_node = '0', ran_node = '0', mpt_hash = '0')
        self.nodes = set()

    def get_mpt(self):
        storage = {}
        trie = MerklePatriciaTrie(storage)
        parm_index = bytes(len(self.chain))
        transactions_list = str(self.transactions)
        transactions_list_hashed = hashlib.sha256((transactions_list).encode()).hexdigest()
        transactions_list_hash_encode = str.encode(transactions_list_hashed)
        mpt_argument = bytes(transactions_list_hash_encode)
```

```
        trie.update(parm_index, mpt_argument)
        old_root_hash = trie.root_hash()
        hash_mpt = format(old_root_hash.hex())
        return hash_mpt

    def get_generating_node(self):
        json = request.get_json()
        nodes_list = json.get('active_permitted_nodes')
        generating_node = random.choice(nodes_list)
        return generating_node

    def hash_generating_node(self):
        target_node = self.get_generating_node()
        target_node_hash = hashlib.sha256((target_node).encode()).hexdigest()
        return target_node_hash

    def get_range_node(self):
        json = request.get_json()
        nodes_list = json.get('active_permitted_nodes')
        range_node = random.choice(nodes_list)
        return range_node

    def hash_range_node(self):
        range_node = self.get_range_node()
        range_node_hash = hashlib.sha256((range_node).encode()).hexdigest()
        return range_node_hash

    def create_block(self, previous_hash, block_time, gen_node, ran_node, mpt_hash
    ):
        block = {'index': len(self.chain) + 1,
                 'gen_node': gen_node,
                 'block_time': block_time,
                 'ran_node': ran_node,
                 'first_id': self.transactions_id[0],
                 'last_id': self.transactions_id[-1],
                 'validation_time': self.validation_time[-1],
                 'previous_hash': previous_hash,
                 'timestamp': str(datetime.datetime.now()),
                 'mpt_hash': mpt_hash,
                 'transactions': self.transactions}
        self.transactions_id = []
        self.transactions = []
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def hash(self, block):
        encoded_block = json.dumps(block, sort_keys = True).encode()
        return hashlib.sha256(encoded_block).hexdigest()

```

```
    def is_chain_valid(self, chain):
        previous_block = chain[0]
        block_index = 1
        while block_index < len(chain):
            block = chain[block_index]
            if block['previous_hash'] != self.hash(previous_block):
                return False
            previous_block = block
            block_index += 1
        return True

    def add_transaction(self, transaction_id, co_banco_orig, co_conta_orig,
    co_banco_dest, co_conta_dest, nu_id_emissor, vr_operacao):
        self.transactions.append({'transaction_id': transaction_id,
                                  'co_banco_orig': co_banco_orig,
                                  'co_conta_orig': co_conta_orig,
                                  'co_banco_dest': co_conta_dest,
                                  'co_conta_dest': co_conta_dest,
                                  'nu_id_emissor': nu_id_emissor,
                                  'vr_operacao': vr_operacao})
        self.transactions_id.append(transaction_id)
        previous_block = self.get_previous_block()
        return previous_block['index'] + 1

    def add_node(self, address):
        parsed_url = urlparse(address)
        self.nodes.add(parsed_url.netloc)

    def replace_chain(self):
        network = self.nodes
        longest_chain = None
        max_length = len(self.chain)
        network_list = list(network)
        for node in network:
            response = requests.get(f'http://{node}/get_chain')
            if response.status_code == 200:
                length = response.json()['length']
                chain = response.json()['chain']
                first_round = requests.get(f'http://{network_list[0]}/is_valid')
                second_round = requests.get(f'http://{network_list[-1]}/is_valid')
                third_round = requests.get(f'http://{network_list[1]}/is_valid')
                if length > max_length and first_round.status_code == 200 and
    second_round.status_code == 200 and third_round.status_code == 200:
                    max_length = length
                    longest_chain = chain
        val_time = str(datetime.datetime.now())
        self.validation_time.append(val_time)
        if longest_chain:
            self.chain = longest_chain
            return True
        return False

```

```
app = Flask(__name__)

node_address = str(uuid4()).replace('-', '')

blockchain = Blockchain()

@app.route('/mine_block', methods = ['GET'])
def mine_block():
    previous_block = blockchain.get_previous_block()
    mpt_hash = blockchain.get_mpt()
    previous_hash = blockchain.hash(previous_block)
    block_time = str(datetime.datetime.now())
    gen_node = blockchain.hash_generating_node()
    ran_node = blockchain.hash_range_node()
    block = blockchain.create_block(previous_hash, block_time, gen_node, ran_node,
     mpt_hash)
    response = {'message': 'ParabÃl'ns, vocÃł minerou um bloco!',
                'index': block['index'],
                'gen_node': block['gen_node'],
                'block_time': block['block_time'],
                'ran_node': block['ran_node'],
                'first_id': block['first_id'],
                'last_id': block['last_id'],
                'validation_time': block['validation_time'],
                'previous_hash': block['previous_hash'],
                'timestamp': block['timestamp'],
                'mpt_hash': block['mpt_hash'],
                'transactions': block['transactions']}
    current_time = str(datetime.datetime.now())
    if block['timestamp'] > current_time:
        return 'O timestamp corrente menor do que o do Ãžltimo bloco da cadeia!',
    400
    return jsonify(response), 200

@app.route('/get_chain', methods = ['GET'])
def get_chain():
    response = {'chain': blockchain.chain,
                'length': len(blockchain.chain)}
    return jsonify(response), 200

@app.route('/is_valid', methods = ['GET'])
def is_valid():
    is_valid = blockchain.is_chain_valid(blockchain.chain)
    if is_valid:
        response = {'message': 'Tudo certo. A cadeia Ãl' vÃąlida!'}
    else:
        response = {'message': 'Temos um problema. A cadeia nÃčo Ãl' vÃąlida!'}
    return jsonify(response), 200

@app.route('/add_transaction', methods = ['POST'])
def add_transaction():
    json = request.get_json()
```

```
    transaction_keys = ['transaction_id', 'co_banco_orig', 'co_conta_orig', '
    co_banco_dest', 'co_conta_dest', 'nu_id_emissor', 'vr_operacao']
195     if not all (key in json for key in transaction_keys):
            return 'Algum elemento da mensagem estÃą faltando', 400
197     index = blockchain.add_transaction(json['transaction_id'], json['co_banco_orig
    '], json['co_conta_orig'], json['co_banco_dest'], json['co_conta_dest'], json['
    nu_id_emissor'], json['vr_operacao'])
        response = {'message': f'Esta transiÃğÃčo serÃą incluida no bloco {index}'}
199     return jsonify(response), 201

201 @app.route('/connect_node', methods = ['POST'])
    def connect_node():
203     json = request.get_json()
        nodes = json.get('active_permitted_nodes')
205     if nodes is None:
            return 'Sem nÃşs!', 400
207     for node in nodes:
            blockchain.add_node(node)
209     response = {'message': 'Todos os nÃşs estÃčo conectados! Quantidade de nÃşs da
     rede: ',
                    'total_nodes': list(blockchain.nodes)}
211     return jsonify(response), 201

213 @app.route('/replace_chain', methods = ['GET'])
    def replace_chain():
215     is_chain_replaced = blockchain.replace_chain()
        if is_chain_replaced:
217         response = {'message': 'Os nÃşs sÃčo diferentes. A maior cadeia ganhou!',
                       'new_chain': blockchain.chain}
219     else:
            response = {'message': 'Tudo certo. A cadeia Ãľ a maior!',
221                     'actual_chain': blockchain.chain}
        return jsonify(response), 200
223
    app.run(host = '0.0.0.0', port = 5000)
```

code/bankchain.py

brucecoin.py

```
# -*- coding: utf-8 -*-

# Importing the libraries
import datetime
import hashlib
import json
from flask import Flask, jsonify, request

import requests
from uuid import uuid4
from urllib.parse import urlparse

class Blockchain:

    def __init__(self):
        self.chain = []
        self.transactions = []
        self.create_block(proof = 1, previous_hash = '0')
        self.nodes = set()

    def create_block(self, proof, previous_hash):
        block = {'index': len(self.chain) + 1,
                 'timestamp': str(datetime.datetime.now()),
                 'proof': proof,
                 'previous_hash': previous_hash,
                 'transactions': self.transactions}
        self.transactions = []
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def proof_of_work(self, previous_proof):
        new_proof = 1
        check_proof = False
        while check_proof is False:
            hash_operation = hashlib.sha256(str(new_proof**2 - previous_proof**2).
    encode()).hexdigest()
            if hash_operation[:5] == '00000':
                check_proof = True
            else:
                new_proof += 1
        return new_proof

    def hash(self, block):
        encoded_block = json.dumps(block, sort_keys = True).encode()
        return hashlib.sha256(encoded_block).hexdigest()

```

```
    def is_chain_valid(self, chain):
        previous_block = chain[0]
        block_index = 1
        while block_index < len(chain):
            block = chain[block_index]
            if block['previous_hash'] != self.hash(previous_block):
                return False
            previous_proof = previous_block['proof']
            proof = block['proof']
            hash_operation = hashlib.sha256(str(proof**2 - previous_proof**2).
    encode()).hexdigest()
            if hash_operation[:5] != '00000':
                return False
            previous_block = block
            block_index += 1
        return True

    def add_transaction(self, sender, receiver, amount):
        self.transactions.append({'sender': sender,
                                  'receiver': receiver,
                                  'amount': amount})
        previous_block = self.get_previous_block()
        return previous_block['index'] + 1

    def add_node(self, address):
        parsed_url = urlparse(address)
        self.nodes.add(parsed_url.netloc)

    def replace_chain(self):
        network = self.nodes
        longest_chain = None
        max_length = len(self.chain)
        for node in network:
            response = requests.get(f'http://{node}/get_chain')
            if response.status_code == 200:
                length = response.json()['length']
                chain = response.json()['chain']
                if length > max_length and self.is_chain_valid(chain):
                    max_length = length
                    longest_chain = chain
        if longest_chain:
            self.chain = longest_chain
            return True
        return False

app = Flask(__name__)

node_address = str(uuid4()).replace('-', '')

blockchain = Blockchain()

@app.route('/mine_block', methods = ['GET'])
```

```
100 def mine_block():
        previous_block = blockchain.get_previous_block()
102     previous_proof = previous_block['proof']
        proof = blockchain.proof_of_work(previous_proof)
104     previous_hash = blockchain.hash(previous_block)
    #   blockchain.add_transaction(sender = node_address, receiver = 'Brucechain',
        amount = 1)
106     block = blockchain.create_block(proof, previous_hash)
        response = {'message': 'ParabÃľns, vocÃł minerou um bloco!',
108                 'index': block['index'],
                    'timestamp': block['timestamp'],
110                 'proof': block['proof'],
                    'previous_hash': block['previous_hash'],
112                 'transactions': block['transactions']}
        return jsonify(response), 200
114
    @app.route('/get_chain', methods = ['GET'])
116 def get_chain():
        response = {'chain': blockchain.chain,
118                 'length': len(blockchain.chain)}
        return jsonify(response), 200
120
    @app.route('/is_valid', methods = ['GET'])
122 def is_valid():
        is_valid = blockchain.is_chain_valid(blockchain.chain)
124     if is_valid:
            response = {'message': 'Tudo certo. A cadeia Ãľ vÃąlida!'}
126     else:
            response = {'message': 'Temos um problema. A cadeia nÃčo Ãľ vÃąlida!'}
128     return jsonify(response), 200

130 @app.route('/add_transaction', methods = ['POST'])
    def add_transaction():
132     json = request.get_json()
        transaction_keys = ['sender', 'receiver', 'amount']
134     if not all (key in json for key in transaction_keys):
            return 'Algum elemento da mensagem estÃą faltando', 400
136     index = blockchain.add_transaction(json['sender'], json['receiver'], json['
        amount'])
        response = {'message': f'Esta transiÃğÃčo serÃą incluida no bloco {index}'}
138     return jsonify(response), 201

140 @app.route('/connect_node', methods = ['POST'])
    def connect_node():
142     json = request.get_json()
        nodes = json.get('nodes')
144     if nodes is None:
            return 'Sem nÃşs!', 400
146     for node in nodes:
            blockchain.add_node(node)
148     response = {'message': 'Todos os nÃşs estÃčo conectados! Quantidade de nÃşs da
         rede: ',
```

```
                    'total_nodes': list(blockchain.nodes)}
    return jsonify(response), 201

@app.route('/replace_chain', methods = ['GET'])
def replace_chain():
    is_chain_replaced = blockchain.replace_chain()
    if is_chain_replaced:
        response = {'message': 'Os nÃşs sÃčo diferentes. A maior cadeia ganhou!',
                    'new_chain': blockchain.chain}
    else:
        response = {'message': 'Tudo certo. A cadeia Ãľ a maior!',
                    'actual_chain': blockchain.chain}
    return jsonify(response), 200

app.run(host = '0.0.0.0', port = 5000)
```

code/brucecoin.py

mpt.py

```
import rlp
from enum import Enum
from Crypto.Hash import keccak

# DependÃłncias:
#
#     cytoolz
#     eth-hash
#     eth-typing
#     eth-utils
#     pycryptodome
#     rlp
#     toolz

class MerklePatriciaTrie:
    def __init__(self, storage, root=None, secure=False):
        self._storage = storage
        self._root = root
        self._secure = secure

    def root(self):
        return self._root

    def root_hash(self):
        if not self._root:
            return Node.EMPTY_HASH
        elif len(self._root) == 32:
            return self._root
        else:
            return keccak_hash(self._root)

    def get(self, encoded_key):
        if not self._root:
            raise KeyError

        if self._secure:
            encoded_key = keccak_hash(encoded_key)

        path = NibblePath(encoded_key)

        result_node = self._get(self._root, path)

        return result_node.data

    def update(self, encoded_key, encoded_value):
        if self._secure:
            encoded_key = keccak_hash(encoded_key)

        path = NibblePath(encoded_key)
```

```
51          result = self._update(self._root, path, encoded_value)
52
53          self._root = result
54
55      def delete(self, encoded_key):
56          if self._root is None:
57              return
58
59          if self._secure:
60              encoded_key = keccak_hash(encoded_key)
61
62          path = NibblePath(encoded_key)
63
64          action, info = self._delete(self._root, path)
65
66          if action == MerklePatriciaTrie._DeleteAction.DELETED:
67              # Trie is empty
68              self._root = None
69          elif action == MerklePatriciaTrie._DeleteAction.UPDATED:
70              new_root = info
71              self._root = new_root
72          elif action == MerklePatriciaTrie._DeleteAction.USELESS_BRANCH:
73              _, new_root = info
74              self._root = new_root
75
76      def _get_node(self, node_ref):
77          raw_node = None
78          if len(node_ref) == 32:
79              raw_node = self._storage[node_ref]
80          else:
81              raw_node = node_ref
82          return Node.decode(raw_node)
83
84      def _get(self, node_ref, path):
85          node = self._get_node(node_ref)
86
87          if len(path) == 0:
88              return node
89
90          if type(node) is Node.Leaf:
91              if node.path == path:
92                  return node
93
94          elif type(node) is Node.Extension:
95              if path.starts_with(node.path):
96                  rest_path = path.consume(len(node.path))
97                  return self._get(node.next_ref, rest_path)
98
99          elif type(node) is Node.Branch:
100             branch = node.branches[path.at(0)]
101             if len(branch) > 0:
```

```
                return self._get(branch, path.consume(1))

        raise KeyError

    def _update(self, node_ref, path, value):
        if not node_ref:
            return self._store_node(Node.Leaf(path, value))

        node = self._get_node(node_ref)

        if type(node) == Node.Leaf:
            if node.path == path:
                node.data = value
                return self._store_node(node)

            common_prefix = path.common_prefix(node.path)
            path.consume(len(common_prefix))
            node.path.consume(len(common_prefix))
            branch_reference = self._create_branch_node(path, value, node.path, node.data)

            if len(common_prefix) != 0:
                return self._store_node(Node.Extension(common_prefix, branch_reference))
            else:
                return branch_reference

        elif type(node) == Node.Extension:

            if path.starts_with(node.path):
                new_reference = self._update(node.next_ref, path.consume(len(node.path)), value)
                return self._store_node(Node.Extension(node.path, new_reference))

            common_prefix = path.common_prefix(node.path)

            path.consume(len(common_prefix))
            node.path.consume(len(common_prefix))

            branches = [b''] * 16
            branch_value = value if len(path) == 0 else b''

            self._create_branch_leaf(path, value, branches)
            self._create_branch_extension(node.path, node.next_ref, branches)

            branch_reference = self._store_node(Node.Branch(branches, branch_value))

            if len(common_prefix) != 0:
                return self._store_node(Node.Extension(common_prefix, branch_reference))
            else:
```

```
149                 return branch_reference
150
151         elif type(node) == Node.Branch:
152             if len(path) == 0:
153                 return self._store_node(Node.Branch(node.branches, value))
154
155             idx = path.at(0)
156             new_reference = self._update(node.branches[idx], path.consume(1),
    value)
157
158             node.branches[idx] = new_reference
159
160             return self._store_node(node)
161
162     def _create_branch_node(self, path_a, value_a, path_b, value_b):
163
164         assert len(path_a) != 0 or len(path_b) != 0
165
166         branches = [b''] * 16
167
168         branch_value = b''
169         if len(path_a) == 0:
170             branch_value = value_a
171         elif len(path_b) == 0:
172             branch_value = value_b
173
174         self._create_branch_leaf(path_a, value_a, branches)
175         self._create_branch_leaf(path_b, value_b, branches)
176
177         return self._store_node(Node.Branch(branches, branch_value))
178
179     def _create_branch_leaf(self, path, value, branches):
180         if len(path) > 0:
181             idx = path.at(0)
182
183             leaf_ref = self._store_node(Node.Leaf(path.consume(1), value))
184             branches[idx] = leaf_ref
185
186     def _create_branch_extension(self, path, next_ref, branches):
187         assert len(path) >= 1, "Path for extension node should contain at least
    one nibble"
188
189         if len(path) == 1:
190             branches[path.at(0)] = next_ref
191         else:
192             idx = path.at(0)
193             reference = self._store_node(Node.Extension(path.consume(1), next_ref)
    )
194             branches[idx] = reference
195
196     def _store_node(self, node):
197         reference = Node.into_reference(node)
```

```
        if len(reference) == 32:
199             self._storage[reference] = node.encode()
        return reference
201
    class _DeleteAction(Enum):
203     DELETED = 1,
        UPDATED = 2,
205     USELESS_BRANCH = 3

207 def _delete(self, node_ref, path):

209     node = self._get_node(node_ref)

211     if type(node) == Node.Leaf:
            if path == node.path:
213             return MerklePatriciaTrie._DeleteAction.DELETED, None
            else:
215             raise KeyError

217     elif type(node) == Node.Extension:
            if not path.starts_with(node.path):
219             raise KeyError

221         action, info = self._delete(node.next_ref, path.consume(len(node.path)
    ))

223         if action == MerklePatriciaTrie._DeleteAction.DELETED:
                return action, None
225         elif action == MerklePatriciaTrie._DeleteAction.UPDATED:
                child_ref = info
227             new_ref = self._store_node(Node.Extension(node.path, child_ref))
                return action, new_ref
229         elif action == MerklePatriciaTrie._DeleteAction.USELESS_BRANCH:
                stored_path, stored_ref = info
231
                child = self._get_node(stored_ref)
233
                new_node = None
235             if type(child) == Node.Leaf:
                    path = NibblePath.combine(node.path, child.path)
237                 new_node = Node.Leaf(path, child.data)
                elif type(child) == Node.Extension:
239                 path = NibblePath.combine(node.path, child.path)
                    new_node = Node.Extension(path, child.next_ref)
241             elif type(child) == Node.Branch:
                    path = NibblePath.combine(node.path, stored_path)
243                 new_node = Node.Extension(path, stored_ref)

245             new_reference = self._store_node(new_node)
                return MerklePatriciaTrie._DeleteAction.UPDATED, new_reference
247
        elif type(node) == Node.Branch:
```

```
249
            action = None
251         idx = None
            info = None
253
            assert len(path) != 0 or len(node.data) != 0, "Empty path or empty
    branch node in _delete"
255
            if len(path) == 0 and len(node.data) == 0:
257             raise KeyError
            elif len(path) == 0 and len(node.data) != 0:
259             node.data = b''
                action = MerklePatriciaTrie._DeleteAction.DELETED
261         else:
                idx = path.at(0)
263
                if len(node.branches[idx]) == 0:
265                 raise KeyError

267             action, info = self._delete(node.branches[idx], path.consume(1))
                node.branches[idx] = b''
269
            if action == MerklePatriciaTrie._DeleteAction.DELETED:
271             non_empty_count = sum(map(lambda x: 1 if len(x) > 0 else 0, node.
    branches))

273             if non_empty_count == 0 and len(node.data) == 0:
                    return MerklePatriciaTrie._DeleteAction.DELETED, None
275             elif non_empty_count == 0 and len(node.data) != 0:
                    path = NibblePath([])
277                 reference = self._store_node(Node.Leaf(path, node.data))

279                 return MerklePatriciaTrie._DeleteAction.USELESS_BRANCH, (path,
     reference)
                elif non_empty_count == 1 and len(node.data) == 0:
281                 return self._build_new_node_from_last_branch(node.branches)
                else:
283                 reference = self._store_node(node)
                    return MerklePatriciaTrie._DeleteAction.UPDATED, reference
285         elif action == MerklePatriciaTrie._DeleteAction.UPDATED:
                next_ref = info
287             node.branches[idx] = next_ref
                reference = self._store_node(node)
289             return MerklePatriciaTrie._DeleteAction.UPDATED, reference
            elif action == MerklePatriciaTrie._DeleteAction.USELESS_BRANCH:
291             _, next_ref = info
                node.branches[idx] = next_ref
293             reference = self._store_node(node)
                return MerklePatriciaTrie._DeleteAction.UPDATED, reference
295
    def _build_new_node_from_last_branch(self, branches):
297     idx = 0
```

```
            for i in range(len(branches)):
                if len(branches[i]) > 0:
                    idx = i
                    break

            prefix_nibble = NibblePath([idx], offset=1)

            child = self._get_node(branches[idx])

            path = None
            node = None

            if type(child) == Node.Leaf:
                path = NibblePath.combine(prefix_nibble, child.path)
                node = Node.Leaf(path, child.data)
            elif type(child) == Node.Extension:
                path = NibblePath.combine(prefix_nibble, child.path)
                node = Node.Extension(path, child.next_ref)
            elif type(child) == Node.Branch:
                path = prefix_nibble
                node = Node.Extension(path, branches[idx])

            reference = self._store_node(node)

            return MerklePatriciaTrie._DeleteAction.USELESS_BRANCH, (path, reference)

    def _prepare_reference_for_usage(ref):
    if isinstance(ref, list):
        return rlp.encode(ref)

    return ref


def _prepare_reference_for_encoding(ref):
    if 0 < len(ref) < 32:
        return rlp.decode(ref)

    return ref


class Node:
    EMPTY_HASH = keccak_hash(rlp.encode(b''))

    class Leaf:
        def __init__(self, path, data):
            self.path = path
            self.data = data

        def encode(self):
            return rlp.encode([self.path.encode(True), self.data])

    class Extension:
```

```
            def __init__(self, path, next_ref):
                self.path = path
                self.next_ref = next_ref

            def encode(self):
                next_ref = _prepare_reference_for_encoding(self.next_ref)
                return rlp.encode([self.path.encode(False), next_ref])

        class Branch:
            def __init__(self, branches, data=None):
                self.branches = branches
                self.data = data

            def encode(self):
                branches = list(map(_prepare_reference_for_encoding, self.branches))
                return rlp.encode(branches + [self.data])

    def decode(encoded_data):
        data = rlp.decode(encoded_data)

        assert len(data) == 17 or len(data) == 2

        if len(data) == 17:
            branches = list(map(_prepare_reference_for_usage, data[:16]))
            node_data = data[16]
            return Node.Branch(branches, node_data)

        path, is_leaf = NibblePath.decode_with_type(data[0])
        if is_leaf:
            return Node.Leaf(path, data[1])
        else:
            ref = _prepare_reference_for_usage(data[1])
            return Node.Extension(path, ref)

    def into_reference(node):
        encoded_node = node.encode()
        if len(encoded_node) < 32:
            return encoded_node
        else:
            return keccak_hash(encoded_node)

class NibblePath:
    ODD_FLAG = 0x10
    LEAF_FLAG = 0x20

    def __init__(self, data, offset=0):
        self._data = data
        self._offset = offset

    def __len__(self):
        return len(self._data) * 2 - self._offset

```

```
    def __repr__(self):
403         return "<NibblePath: Data: 0x{}, Offset: {}>".format(self._data.hex(),
    self._offset)

405     def __str__(self):
        return '<Hex 0x{} | Raw {}>'.format(self._data.hex(), self._data)
407
    def __eq__(self, other):
409         if len(self) != len(other):
                return False
411
            for i in range(len(self)):
413             if self.at(i) != other.at(i):
                    return False
415
            return True
417
    def decode_with_type(data):
419         is_odd_len = data[0] & NibblePath.ODD_FLAG == NibblePath.ODD_FLAG
            is_leaf = data[0] & NibblePath.LEAF_FLAG == NibblePath.LEAF_FLAG
421
            offset = 1 if is_odd_len else 2
423
            return NibblePath(data, offset), is_leaf
425
    def decode(data):
427         return NibblePath.decode_with_type(data)[0]

429     def starts_with(self, other):
            if len(other) > len(self):
431             return False

433         for i in range(len(other)):
                if self.at(i) != other.at(i):
435                 return False

437         return True

439     def at(self, idx):
            idx = idx + self._offset
441
            byte_idx = idx // 2
443         nibble_idx = idx % 2

445         byte = self._data[byte_idx]

447         nibble = byte >> 4 if nibble_idx == 0 else byte & 0x0F

449         return nibble

451     def consume(self, amount):
            self._offset += amount
```

```
453         return self
455     def _create_new(path, length):
            bytes_len = (length + 1) / 2
457         data = []
459         is_odd_len = length % 2 == 1
            pos = 0
461
            if is_odd_len:
463             data.append(path.at(pos))
                pos += 1
465
            while pos < length:
467             data.append(path.at(pos) * 16 + path.at(pos + 1))
                pos += 2
469
            offset = 1 if is_odd_len else 0
471
            return NibblePath(data, offset)
473
        def common_prefix(self, other):
475         least_len = min(len(self), len(other))
            common_len = 0
477         for i in range(least_len):
                if self.at(i) != other.at(i):
479                 break
                common_len += 1
481
            return NibblePath._create_new(self, common_len)
483
        def encode(self, is_leaf):
485         output = []
487         nibbles_len = len(self)
            is_odd = nibbles_len % 2 == 1
489
            prefix = 0x00
491         prefix += self.ODD_FLAG + self.at(0) if is_odd else 0x00
            prefix += self.LEAF_FLAG if is_leaf else 0x00
493
            output.append(prefix)
495
            pos = nibbles_len % 2
497
            while pos < nibbles_len:
499             byte = self.at(pos) * 16 + self.at(pos + 1)
                output.append(byte)
501             pos += 2
503         return bytes(output)
```

```
505     class _Chained:
            def __init__(self, first, second):
507             self.first = first
                self.second = second
509
            def __len__(self):
511             return len(self.first) + len(self.second)

513         def at(self, idx):
                if idx < len(self.first):
515                 return self.first.at(idx)
                else:
517                 return self.second.at(idx - len(self.first))

519     def combine(self, other):
            chained = NibblePath._Chained(self, other)
521         return NibblePath._create_new(chained, len(chained))

523     def keccak_hash(data):
            keccak_hash = keccak.new(digest_bits=256)
525         keccak_hash.update(data)
            return keccak_hash.digest()
```

code/mpt.py