Simulando o Hedera Hashgraph

Guilherme de L. Ferreira[1], **Pedro H. F. Von Zuben**[1],
Raphael de C. Mesquita[1], **Yuri V. C. de M. A. da Silveira**[1]

[1]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)

DRE: 121143034, DRE: 119055699, DRE: 118020104, DRE: 119104804

Resumo.

1. Introdução

1.1. *Ledger*

Um *ledger* é essencialmente um registro que documenta todas as transações dentro de um sistema, preservando informações relevantes sobre cada uma delas. Embora o conceito de ledger possa ser aplicado a qualquer tipo de registro organizado, ele é mais comumente associado ao registro de transações financeiras de uma empresa, sendo nesse caso conhecido como livro-razão.

Historicamente, os *ledgers* começaram como registros físicos em papel onde cada transação era cuidadosamente anotada. Esses livros-razão eram importantes para a contabilidade de empresas, permitindo que todas as movimentações financeiras fossem monitoradas e verificadas. Isso também ajudava a garantir que os registros financeiros fossem precisos e completos. Com o avanço da tecnologia, os *ledgers* evoluíram para sistemas eletrônicos, permitindo o armazenamento e a gestão de dados em escala muito maior e com maior eficiência. Entretanto, esses sistemas ainda tinham suas limitações, como a necessidade de confiança em um único ponto de controle e a vulnerabilidade a falhas e fraudes.

1.2. *Distributed Ledger*

Um *distributed ledger* é uma versão mais avançada do *ledger* tradicional, com algumas características-chave adicionais. Por exemplo, em vez de um único ponto central onde todas as transações são registradas, os registros são distribuídos entre múltiplos nós (computadores) em uma rede. Consequentemente, não há uma autoridade central controlando o *ledger*. Além disso, a arquitetura descentralizada reduz a dependência de um ponto único de falha e melhora a segurança e resiliência do sistema.

Paralelamente, como os registros são distribuídos entre muitos nós, todos eles mantêm uma cópia do ledger e devem concordar sobre o estado das transações. Este consenso é alcançado por meio de algoritmos de consenso, como *Proof of Work* (PoW), *Proof of Stake* (PoS), *Practical Byzantine Fault Tolerance* (PBFT) e entre outros. Esses algoritmos garantem que todos os nós na rede validem e concordem com as transações, tornando o sistema mais resistente a falhas e manipulações.

Uma vez que uma transação é registrada em um *distributed ledger*, ela é geralmente imutável. Em outras palavras, qualquer tentativa de alterar uma transação registrada é facilmente detectada porque todas as cópias do ledger em todos os nós da rede

devem concordar sobre o estado das transações. A imutabilidade é reforçada pelo uso de hashes criptográficos, que vinculam blocos de transações em uma cadeia (*Blockchain*), tornando qualquer alteração posterior perceptível e invalidando a cadeia subsequente.

A segurança em um *distributed ledger* é garantida por meio de criptografia e algoritmos de hashing. De modo que, cada transação é assinada digitalmente e os dados são protegidos contra adulteração. Ademais, o uso de criptografia assimétrica garante que apenas usuários autorizados possam iniciar transações e acessar dados específicos no *ledger*. No geral, *distributed ledgers* fornecem um registro transparente de todas as transações, permitindo que qualquer participante da rede verifique as transações a qualquer momento. Esta transparência aumenta a confiança no sistema e facilita auditorias independentes.

1.3. *Blockchain*

A *Blockchain* é uma tecnologia de livro-razão distribuído que se tornou sinônimo de segurança e transparência em registros digitais. De um modo geral, caracteriza-se por uma sequência de blocos de dados, onde cada bloco contém uma lista de transações e está ligado ao bloco anterior através de um *hash* criptográfico, formando uma cadeia ininterrupta e imutável. Em vez de um único ponto central onde todas as transações são registradas, os registros são distribuídos entre múltiplos nós em uma rede *peer-to-peer* (P2P). Cada participante da rede mantém uma cópia desse *ledger*, garantindo tanto a auditabilidade quanto a descentralização. As transações são agrupadas em blocos, e cada bloco contém um *hash* que o liga irreversivelmente ao bloco anterior, criando uma cadeia de blocos (*Blockchain*). Esta ligação dificulta a alteração de qualquer registro anterior sem que isso seja detectado, assegurando a imutabilidade da cadeia.

A segurança na *Blockchain* é garantida por criptografia e consenso distribuído. Cada bloco contém um *hash* criptográfico do bloco anterior, protegendo a integridade e autenticidade das transações. No entanto, *Blockchain*s baseadas em PoW estão sujeitas a, por exemplo, um "*51% attack*", onde um ator mal-intencionado controla mais de 50% da potência de *hash* da rede. Em sistemas baseados em PoS, o problema do "nada em jogo"pode surgir, enfraquecendo a segurança da rede. Devido à sua natureza transparente e imutável, a *Blockchain* é altamente auditável. Qualquer transação realizada na rede pode ser rastreada e verificada por qualquer participante, garantindo a confiança e a integridade do sistema. Além disso, a *Blockchain* permite que as transações sejam feitas de forma pseudônima, onde os participantes são identificados por endereços públicos, não por informações pessoais.

A criação de novos blocos é regida por algoritmos de consenso, que garantem que todos os participantes da rede concordem sobre a validade das novas transações e a ordem em que elas são registradas no *ledger*. Os mecanismos de consenso incluem PoW, que requer que os mineradores resolvam problemas matemáticos complexos para validar as transações e criar novos blocos, e PoS, que seleciona validadores com base na quantidade de criptomoeda que eles possuem e estão dispostos a "apostar"como garantia. Cada método tem seus próprios benefícios e desvantagens em termos de segurança, eficiência e consumo de energia.

Paralelamente, a escalabilidade é um dos principais desafios da *Blockchain*. De modo que, à medida que a rede cresce, a capacidade de processar um grande volume de

transações de maneira rápida e eficiente pode ser limitada. Para mitigar esses problemas, estão sendo desenvolvidas soluções como a *Lightning Network*, uma rede de segunda camada que permite transações rápidas e de baixo custo para *Bitcoin*, e o sharding para *Ethereum*, uma técnica que divide a rede em shards menores, permitindo o processamento paralelo de transações. Além disso, sidechains, cadeias paralelas que operam independentemente da cadeia principal, mas são conectadas a ela, são exploradas como solução.

*Blockchain*s baseadas em PoW consomem uma quantidade significativa de energia devido ao processo de mineração intensivo. Alternativas como PoS visam reduzir este consumo, mas enfrentam seus próprios desafios. A latência nas transações e a complexidade de realizar atualizações ou modificações significativas na rede (*hard forks*) são problemas adicionais que a *Blockchain* enfrenta.

1.4. *Hashgraph*

O *Hashgraph*, proposto pela Hedera, é uma tecnologia de *Distributed Ledger* com a proposta de superar as limitações da *blockchain* tradicional, oferecendo melhorias significativas em vários aspectos. A segurança é garantida pela resistência a falhas bizantinas assíncronas (aBFT), permitindo que a rede mantenha sua integridade mesmo na presença de atores mal-intencionados. Tornando o *Hashgraph* robusto contra ataques que tentam manipular um número significativo de nós. A aBFT também assegura que o sistema possa tolerar até um terço dos nós sendo comprometidos sem afetar o consenso ou a precisão dos dados.

Um dos principais problemas das *blockchains* tradicionais é a escalabilidade, pois enfrentam dificuldades em processar grandes volumes de transações rapidamente à medida que a rede cresce. O *Hashgraph*, por outro lado, é projetado para ser altamente escalável, permitindo milhares de transações por segundo (TPS). Isso é possível devido à sua estrutura de grafo acíclico dirigido (DAG), que processa transações em paralelo em vez de sequencialmente.

Assim como a *blockchain*, o *Hashgraph* garante a auditabilidade ao manter um registro imutável de todas as transações, que podem ser verificadas por qualquer participante da rede. A imutabilidade é reforçada pela estrutura DAG, que assegura que qualquer alteração retroativa nos dados seja detectada imediatamente por todos os nós participantes.

O *Hashgraph* utiliza um método de consenso diferente das *blockchains* tradicionais. Em vez de depender de algoritmos de consenso, ele usa a combinação de "*gossip about gossip*"e "*virtual voting*"para alcançar o consenso de forma eficiente e rápida. Isso elimina a necessidade de um consumo excessivo de energia e reduz o tempo necessário para confirmar as transações.

Embora a *blockchain* enfrente desafios como ataques e escalabilidade limitada, o *Hashgraph* tenta mitigar esses problemas com seu design inovador. A estrutura DAG e os métodos de consenso evitam o congestionamento da rede e tornam-na menos vulnerável a ataques. No entanto, como qualquer tecnologia emergente, o *Hashgraph* ainda enfrenta desafios, incluindo a necessidade de adoção ampla para maximizar sua segurança e eficiência, questões regulatórias e concorrência com outras tecnologias de ledger distribuído. A eficiência a longo prazo e a adoção do *Hashgraph* continuam a ser avaliadas à medida que mais projetos e aplicações são desenvolvidos sobre essa plataforma.

2. Trabalhos Relacionados

Esta seção revisa alguns dos principais artigos que foram usados de base para o entendimento das tecnologias de distributed ledger, suas utilidades e seus desafios.

Inicialmente foi utilizado de base a survey [Hasanova et al. 2019] para um melhor entendimento dos conceitos que cercam a blockchain e os desafios para a adoção em larga escala da tecnologia, desde os problemas de segurança em cada versão quanto os problemas de escalabilidade.

A partir desse artigo, e o entendimento dos dilemas da escalabilidade e privacidade dentro das blockchains, foram pesquisadas alternativas para as implementações mais famosas da tecnologia como a Bitcoin e a Ethereum. Primeiramente, foram pesquisadas implementações alternativas do modelo DLT das blockchains, como no artigo [Salim et al. 2024] em que é proposto um modelo baseado em Ethereum mais escalavel e privado para o compartilhamento e processamento de dados médicos.

Depois, foram pesquisadas DLTs alternativas à blockchain e assim foi descoberta a tecnologia Hashgraph e suas particularidades. Uma vez decidido o enfoque do trabalho na tecnologia Hashgraph e a vontade de realizar uma implementação dela, foram usados como norte os documentos base para a criação do Hashgraph e da Hedera [Baird 2016], [Baird et al. 2019] que serviram para entender como se estabelecem as redes nessa tecnologia, como funciona a comunicação nelas e como é estabelecido o consenso para o funcionamento da DLT.

3. Visão geral do Hashgraph e Implementação

O *hashgraph* é uma estrutura de dados baseada em DAG. Diferentemente de uma Blockchain, que organiza transações em blocos lineares, o *hashgraph* registra transações (ou apenas a sincronização de informações usando o protocolo "*Gossip about Gossip*") como eventos que são representados como vértices em um grafo. Cada evento contém:

- **Carga útil (*payload*) de novas transações:** Detalhes sobre qualquer nova transação que o criador do evento desejou registrar naquele momento. Esses detalhes podem incluir informações como quem enviou a transação, quem a recebeu, o valor transferido, etc. É importante lembrar que a presença de uma carga útil é opcional. Nem todos os eventos precisam registrar transações; alguns podem apenas registrar sincronizações de informações.
- **Carimbo da data e hora da criação do evento (*timestamp*):** Indica o momento exato em que o evento foi criado. Ele é crucial para ordenar os eventos temporalmente e para resolver conflitos de transação.
- ***Hash* do evento:** Cada evento contém um hash único, calculado a partir de seu conteúdo. Este hash é usado para determinar a posição relativa do evento no *hashgraph* e para verificar a integridade do evento.
- **Hashes de dois eventos anteriores:** Cada evento em um *hashgraph* referência dois eventos anteriores, que são necessários para saber se houve um fork e para o funcionamento do protocolo "*Gossip about Gossip*":

  - **Auto-pai (*Self-parent*):** um evento anterior criado pelo mesmo nó que está criando o novo evento.
  - **Outro-pai (*Other-parent*):** um evento anterior criado por um nó diferente, com o qual o criador do novo evento trocou informações.

- **Assinatura digital do criador do evento:** Garante a autenticidade do evento, identificando o nó que criou o evento na rede hashgraph.

3.1. *Gossip about Gossip*

O Hashgraph utiliza um protocolo de fofoca sobre fofoca (gossip about gossip) para a propagação de informações. Este protocolo é uma extensão do protocolo de fofoca e funciona da seguinte maneira:

- **Fofoca:** Quando um membro da rede, digamos ”A”, cria uma nova transação, ele seleciona aleatoriamente outro membro, digamos ”B”, e compartilha todas as informações que possui, incluindo a nova transação. Após esse compartilhamento inicial, tanto ”A”quanto ”B”irão, cada um, selecionar aleatoriamente novos membros da rede para compartilhar as informações recebidas. Esse processo de compartilhamento continua, com cada membro que recebe as informações escolhendo novos membros aleatoriamente para repassar a informação. Esse método de propagação, conhecido como ”fofoca”, permite que as informações se espalhem exponencialmente pela rede. Em pouco tempo, todos os membros da rede terão conhecimento da nova transação criada pelo membro ”A”.

- **Fofoca sobre Fofoca :** Cada evento no hashgraph registra, além de seu próprio hash, os hashes de dois eventos anteriores: um do próprio membro da rede, digamos ”A”e outro do membro com quem ele compartilhou informações, digamos ”B”. Isso é essencial porque, além de compartilhar transações, os membros também compartilham informações sobre quem conversou com quem e em que ordem. Esse processo, conhecido como ”Fofoca sobre Fofoca”, cria um registro detalhado de todas as interações na rede. Esse registro é fundamental para a ”votação virtual”(**virtual voting**), que utiliza essas informações para alcançar o consenso de maneira rápida e eficiente. Com todos os membros possuindo uma visão completa e atualizada do hashgraph, a rede pode determinar a ordem das transações de forma descentralizada, segura e precisa

3.2. *Virtual Voting*

No contexto dos algoritmos tradicionais de consenso tolerantes a falhas bizantinas (BFT) sem líder, o consenso é frequentemente alcançado através de múltiplas rodadas de votação. Cada membro da rede envia seu voto para cada outro membro, resultando em um grande número de mensagens trocadas. Esse processo pode ser muito custoso em termos de largura de banda e tempo, especialmente à medida que o número de membros da rede cresce.

O Hashgraph, no entanto, evita esse overhead utilizando a ”votação virtual”. Nesta abordagem, cada membro da rede mantém uma cópia do *hashgraph* completo (utilizando

Figura 1. Taxas ao longo das sequências para cada N

o protocolo de “Fofoca sobre Fofoca”) . A partir desta estrutura de dados compartilhada, os membros podem calcular os votos que teriam sido enviados, sem realmente enviá-los.

A consistência entre as cópias do *hashgraph* de diferentes membros é garantida porque, eventualmente, todos os membros aprenderão sobre todos os eventos através da “Fofoca sobre Fofoca”. No entanto, pode haver diferenças temporárias devido a eventos recentes que ainda não foram compartilhados com todos os membros. Essas pequenas discrepâncias são resolvidas à medida que o protocolo de “Fofoca sobre Fofoca” continua a funcionar.

4. Simulações e Testes

A partir do código implementado realizamos algumas simulações a fim de realizar algumas métricas. Dentre elas, trataremos especificamente de:

- **Taxa e taxa média de criação de eventos:** Eventos são criados a cada fofoca realizada, assim conseguimos medir, de certa forma, a vazão do *hashgraph*, verificando quantos eventos são criados por segundo. Com isso, conseguimos, também, analisar a escalabilidade da implementação proposta, aumentando ou diminuindo o número de usuários.

- **Média de eventos criados até que haja consenso:** Uma vez que um consenso é decido, possíveis transações são realizadas. Assim, com essa média conseguimos estipular uma taxa de transferência máxima para estas transações. Da mesma forma que no item anterior, conseguimos também analisar a questão de escalabilidade.

Destacamos que nossa implementação ela não é, de fato, paralela. Escolhemos lidar com esta simulação com as fofocas sendo feitas de forma sequencial por conta da dificuldade e tempo extra para paralelizar a aplicação. Portanto, nossa análise será feita a partir de um *hashgraph* ”piorado”, o que não descaracteriza o propósito deste trabalho.

4.1. Parâmetros

Para ambos testes, variamos o número de usuários N na rede. Detalhadamente, $N = 2$, $N = 4$, $N = 6$ e $N = 8$.

Além disso, em todas as simulações, precisamos focar as medições no *hashgraph* de um usuário específico. Isso é necessário, pois o grafo, apesar de convergir a um comum entre todos os usuários, a convergência não é instantânea. Assim, escolhemos, para todos os casos, o primeiro usuário da rede.

4.2. Taxa e taxa média de criação de eventos

Para a taxa de criação de eventos, calculamos, na verdade, o número de fofocas realizadas por segundo. Neste caso, medimos o intervalo de tempo Δt necessário para cem fofocas serem feitas, ou seja, obtemos o valor da taxa $T_e = \frac{100}{\Delta t}$. Além disso, a cada cem fofocas, definimos uma sequência de fofocas. Portanto, a primeira sequência engloba as cem primeiras fofocas, a segunda sequência, engloba as cem fofocas depois da primeira sequência, e assim por diante. Assim, na figura abaixo, observamos a taxa T_{e_i} para cada sequência $1 \leq i \leq 10$ de fofocas, para cada número de usuários N.

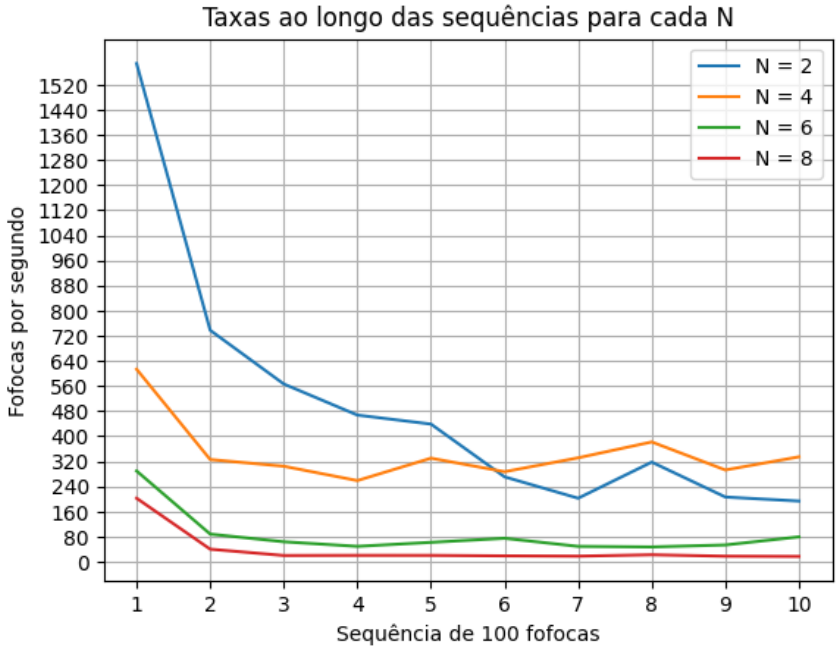

Figura 2. Taxas ao longo das sequências para cada N

Além disso, calculamos também a taxa média T_m da criação desses eventos, em dez sequências, a partir da expressão

$$T_m = \frac{1}{10} \sum_{i=1}^{10} T_{e_i}$$

Assim, a Figura x abaixo apresenta o gráfico com as taxas médias para cada número N de usuários.

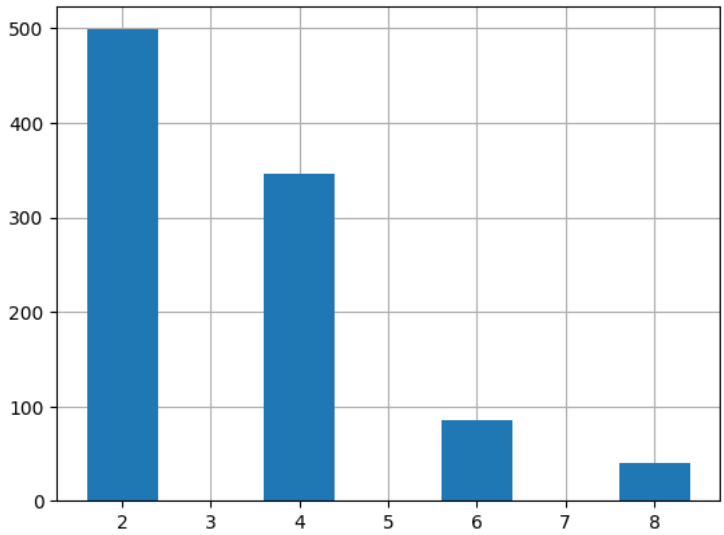

Figura 3. Taxas médias para cada N

4.3. Média de eventos criados até que haja consenso

Neste teste, contaremos os eventos criados até que um consenso é alcançado. Podemos definir como C_i o i-ésimo consenso e E_i o número de eventos criados entre C_{i-1} e C_i. Assim, a Figura x abaixo, apresenta o gráfico $C_i \times E_i$, para cada número N de usuários, com $1 \leq i \leq 10$.

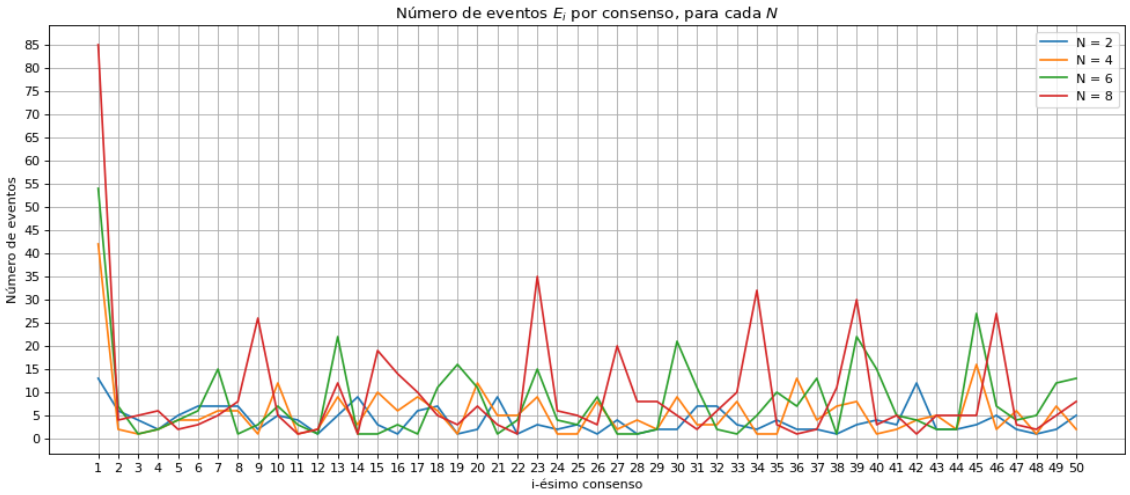

Figura 4. Número de eventos E_i por consenso, para cada N

Enfim, calculamos essa média de forma análoga ao item anterior. Ou seja,

$$M_e = \frac{1}{10} \sum_{i=1}^{10} E_i$$

Na Figura x podemos observar o gráfico com os valores das médias M_e para cada número N de usuários.

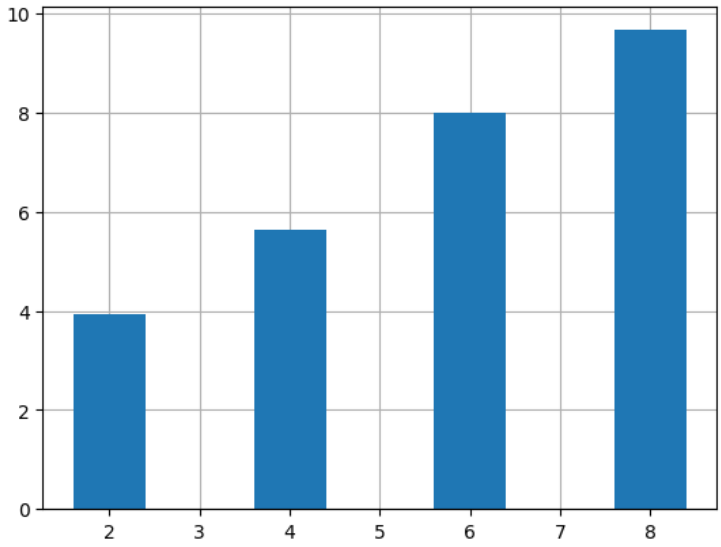

Figura 5. Média de eventos criados M_e, para cada N

5. Discussões

Aqui discutiremos problemas e vantagens encontrados e possíveis outros resultados obtidos a partir das simulações realizadas

6. Conclusão

Seção para resumir tudo o que foi discutido e sintetizar em uma ideia coesa e concisa.

7. Referências

Baird, L. (2016). The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, 34:9–11.

Baird, L., Harmon, M., and Madsen, P. (2019). Hedera: A public hashgraph network & governing council. *White Paper*, 1(1):9–10.

Hasanova, H., Baek, U.-j., Shin, M.-g., Cho, K., and Kim, M.-S. (2019). A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. *International Journal of Network Management*, 29(2):e2060.

Salim, M. M., Yang, L. T., and Park, J. H. (2024). Privacy-preserving and scalable federated blockchain scheme for healthcare 4.0. *Computer Networks*, 247:110472.