Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Protocolo de comunicação usando blockchain para dispositivos IoT

Eduardo Castro Serra

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Daniel Chaves Café

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Protocolo de comunicação usando blockchain para dispositivos IoT

Eduardo Castro Serra

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Daniel Chaves Café (Orientador)
ENE/UnB

Prof. Dr. João José Costa Gondim
CIC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
ENE/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 27 de maio de 2021

Dedicatória

Eu dedico esse trabalho aos meus pais, os quais sempre me ajudam em tudo que eu precisar e me incentivam a alcançar os meus objetivos, à minha irmã e também aos meus amigos Clara, João e Matheus, que são meus amigos desde o início da faculdade e sempre estão presentes em todas as situações.

Agradecimentos

Eu agradeço à minha família, especialmente meus pais, os quais me deram todo o apoio necessário, sem eles esse trabalho não teria sido realizado.

Agradeço aos meus amigos, especialmente Clara, João e Matheus. Todas as experiências e a convivência que eu tive com eles durante esses anos foram essenciais.

Também agradeço aos meus professores que durante os últimos 5 anos me ensinaram todo o conteúdo necessário para que eu pudesse fazer esse trabalho. Agradeço especialmente ao professor e orientador desse projeto Daniel Café, o qual desde antes do início do trabalho sempre demonstrou ser muito solícito e contribuiu ativamente para que eu conseguisse terminá-lo.

Resumo

A cada ano, bilhões de novos dispositivos IoT são produzidos e eles estão presentes em diferentes setores, nas indústrias, nas casas, nas cidades e novas tecnologias, como o 5G, incentivam cada vez mais o uso desses aparelhos. Ao mesmo tempo, devido à falta de padronização e o baixo poder de processamento, novas falhas de segurança são descobertas nesses dispositivos. Nesse contexto, este trabalho apresenta um protocolo para comunicação usando blockchain capaz de identificar a violação de dados durante a transmissão a fim de garantir a integridade das informações geradas por dispositivos IoT. Esse protocolo possui 4 estruturas que interagem com uma blockchain: dispositivos (IoT), nós mineradores (responsáveis por colocar os dados no formato apropriado para armazenar na blockchain), nós centrais (os quais armazenam a blockchain) e atores (aplicativos e sistemas), que são usadas para obter dados a partir de dispositivos IoT, armazená-los em uma blockchain e enviar para aplicativos e sistemas. Para avaliar o protocolo, um man-in-the-middle foi introduzido no envio dos dados dos nós centrais para os atores e, caso ele não fosse capaz de alterar as informações abaixo de um tempo limite, seria possível detectar a invasão. Pelos resultados obtidos, foi possível identificar alterações nos dados feitas por um invasor, sendo que a diferença de tempo de um envio sem modificações para um com modificações aumentava à medida que a blockchain crescia, em alguns casos, um envio com um man-in-the-middle demorou mais de 700 segundos para ser recebido por um ator, enquanto um envio sem invasão com os mesmos dados demorou menos de 1 segundo. Dois problemas observados pelos resultados foram o alto poder computacional necessário aos nós mineradores e o tempo elevado para armazenar os dados na blockchain. Em trabalhos futuros, pode-se tentar diminuir o poder computacional necessário e o tempo para armazenamento trocando o algoritmo de consenso da blockchain e também pode-se combinar o protocolo com outros.

Palavras-chave: IoT, Blockchain, Man-in-the-middle

Abstract

Billions of IoT devices are manufactured each year and they are used in industries, in houses, in cities and tecnologies such as 5G Internet impacts in adoption of these devices. However, the lack of standards and the low computational power favor new vulnerabilities to be discovered. This work presents a blockchain protocol for communication that can identify attacks in integrity of data created by IoT devices. This protocol have 4 structures: IoT devices, miner nodes (convert data to format that can be stored in blockchain), hub nodes (store data in blockchain) and applications (or systems). These structures are required to extract data from IoT devices, store them in blockchain and send to applications and systems. A man-in-the-middle (MITM) have been inserted between hub nodes and an application to evaluate if the protocol could detect an attack. An attack can be prevented if MITM can't modify data below a timeout. The results have revealed that changes in data made by an attacker are succesfully detected, because in some tests the time difference between data transmitted with and without modification have been higher than 700 seconds. The results have also shown that the protocol needs a high computational power in miner nodes and high amount of time to store data in blockchain. In future works this high computational power perhaps can be reduced changing the blockchain consensus algorithm and they can try to combine this protocol with others.

Keywords: IoT, Blockchain, Man-in-the-middle

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Atualmente o mercado de Internet das Coisas (IoT) está em notável crescimento, pela redução de custos, melhoria de desempenho dos aparelhos e também melhorias nas tecnologias de redes de internet. Os dispositivos IoT começam a aparecer mais frequentemente em casas, a chamada Smart Home, composta de luzes inteligentes como o Philips Hue, controladas por voz ou por celulares, câmeras de segurança com alertas automáticos, travas e cadeados com desbloqueio por reconhecimento facial, dispositivos com comandos de voz como Google Home e Amazon Echo que já venderam mais de 50 milhões de unidades cada, além de eletrodomésticos, como geladeiras, que realizam compras e gerenciam os alimentos. Um sistema de luzes inteligentes pode receber informações sobre horário para ligar e desligar, quantidade de luz que deve fornecer, condições de iluminação no ambiente. Com esses elementos, ele consegue adaptar as luzes de acordo com a necessidade. Além desses dispositivos usados por usuários finais, existem diferentes minicomputadores e microcontroladores utilizados para o desenvolvimento de soluções para IoT, entre eles, destacam-se o Raspberry Pi (atualmente na versão 4) e o Arduino Uno (com diferentes versões entre elas o Rev3) que possuem um preço baixo e suporte tanto dos criadores quanto de comunidades de desenvolvedores que facilitam a entrada de novatos. Também há várias plataformas para facilitar o desenvolvimento para IoT como Google Cloud IoT, Amazon AWS IoT Core e Microsoft Azure IoT Hub. Também novas tecnologias, como o 5G, estão sendo criadas para facilitar a comunicação dos aparelhos e permitir o crescimento de casas, cidades e indústrias inteligentes e conectadas com a internet.

Entretanto, ao mesmo tempo que ocorre um rápido crescimento nessa área, novas falhas de segurança são descobertas e os dispositivos IoT, por serem desenvolvidos com hardware de baixo poder de processamento, muito frequentemente são alvos fáceis para invasores. Indivíduos maliciosos podem usar das falhas de segurança de tais dispositivos para ter acesso à dados privados das pessoas. Deve-se tomar cuidado adicional quando utilizamos dispositivos com microfones e câmeras, pois estes podem gerar dados direta-

mente relacionados à privacidade. Outros dispositivos menos invasivos, como controle de luminosidade, reabastecimento de alimento na geladeira, podem gerar dados sobre os hábitos das pessoas, o que pode ser aproveitado por indivíduos maliciosos. Algumas falhas notáveis foram identificadas nos últimos anos e inclusive foram utilizadas, entre elas: a invasão do sistema de entretenimento de um Jeep Cherokee em 2015 que exigiu o recall de mais de 1 milhão de veículos desse modelo; a invasão de uma geladeira inteligente RF28HMELBSR da Samsung, também em 2015, que permitiu acesso às credenciais de login de contas do Gmail associadas às geladeiras; aparelhos médicos como marca-passos e desfibriladores da St. Jude Medical que possuíam uma vulnerabilidade permitiam a invasores modificar comandos dos dispositivos remotamente. Além dessas falhas, houve um problema de segurança em 2016, no qual dispositivos IoT conectados na internet eram invadidos, por possuir usuário e senha padrões para autenticação, e eram infectados com um programa malicioso para serem utilizados em um ataque de negação de serviço (DDoS) ao provedor Dyn. Esse ataque ficou conhecido como Mirai Botnet e prejudicou o funcionamento de serviços como Twitter, the Guardian, Netflix, Reddit e CNN.

Essas situações podem ocorrer devido à falta de padronização dos diferentes aparelhos de IoT e de como eles devem se comunicar de forma a garantir segurança e privacidade dos dados ou também pela possibilidade de serem usados em ambientes não controlados, nos quais podem ocorrer interações com vários outros dispositivos ou com distintas pessoas. Além disso, normalmente esses dispositivos possuem recursos limitados e implementações de segurança se tornam um grande desafio.

Neste trabalho, estamos interessados em desenvolver uma tecnologia para aprimorar e tornar segura a transferência de dados entre dispositivos IoT. Para isso, serão discutidos métodos para garantir a segurança na transmissão de dados utilizando modelos de Blockchain.

Blockchain pode ser definida simplificadamente como uma tecnologia para envio e recebimento de dados de forma descentralizada, ou seja, duas partes podem se comunicar sem a necessidade de um terceiro. Esses dados são registrados em transações imutáveis chamadas de blocos o que explica o termo "block". À medida que novas transações vão sendo criadas, os blocos são conectados, isso cria a ideia de que os blocos estão ligados em uma corrente, por isso o termo "chain". Uma blockchain pode ser criada de diferentes modos, contudo qualquer pessoa com acesso a essa blockchain pode rastrear todos os detalhes das transações e, com isso, é possível com essa comunicação descentralizada ter confiança nos dados que uma pessoa está enviando à outra. Esse conceito sobre a blockchain surgiu em 2008 no artigo Bitcoin: A Peer-to-Peer Electronic Cash System[1] e seu primeiro uso foi em 2009 quando a Bitcoin foi lançada em código aberto. A Bitcoin é uma criptomoeda descentralizada que, usando a blockchain, permitiu transações sem a presença de interme-

diários e que podem ser verificadas por todos os usuários. Normalmente, moedas, as quais são chamadas de fiduciárias, permitem transações por terem sido emitidas por governos ou bancos centrais e possuírem um valor associado a economia de um país, entre essas moedas há o dólar, o euro, o real. Já uma criptomoeda não possui essa emissão por uma entidade administradora e, por isso, não há a presença de intermediários. Posteriormente diversas outras criptomoedas surgiram, entre elas o Ethereum, que não é apenas uma criptomoeda, mas também uma plataforma para a execução de aplicações descentralizadas e contratos inteligentes (são contratos realizados entre duas ou mais pessoas, que, em vez de escritos em papel com linguagem jurídica, são implementados com linguagem de programação para serem executados em computadores). Hoje, a blockchain é usada para diferentes aplicações não apenas ligadas a transações financeiras como as criptomoedas. Alguns exemplos de uso são: a criação de certificados digitais que verificam o status de saúde de passageiros de aeroportos em relação ao SARS-CoV-2, serviço de transferência de dados sensíveis entre médicos e pacientes, métodos de autenticação em servidores sem necessidade de senha, registros de transações e transporte de mercadorias por empresas de logística.

Portanto, devido à possibilidade de usar a blockchain para aplicações que não são financeiras, além das características de confiança e imutabilidade, será proposto um protocolo de comunicação seguro capaz de identificar alterações dos dados no meio da transmissão e detectar interceptação por tempo de computação/envio utilizando os modelos de blockchain que serão discutidos. Para avaliar a segurança na transmissão dos dados, será verificado se é possível identificar um ataque de man-in-the-middle (ataque em que um invasor intercepta e até mesmo modifica os dados enviados na comunicação entre duas partes) entre a estrutura responsável por armazenar a blockchain e um aplicativo ou sistema que utilizará os dados armazenados e também se é possível impedir que o invasor altere o funcionamento de um dispositivo. Para isso será utilizado como métrica o tempo necessário para realizar uma transmissão de dados quando os dados são enviados no formato de blocos. Também será avaliado como a blockchain interfere na geração e coleta de dados pelos dispositivos verificando quanto tempo é necessário desde a obtenção dos dados até o envio para aplicativos e sistemas que utilizem os dados.

Capítulo 2

Revisão Bibliográfica

2.1 Microcontrolador

Um microcontrolador é um dispositivo de circuito integrado, usado para controlar outras partes de um sistema, por meio de uma unidade de microprocessamento (MPU), memória e periféricos. [2] São otimizados para aplicações específicas, repetitivas e com baixo consumo de energia. Eles interagem com componentes digitais, analógicos e eletromecânicos.

Alguns exemplos de microcontroladores são Arduino, ESP32, STM32 e eles possuem limitação na capacidade de processamento. Como pode ser observado na Tabela 2.1, é comum a ausência de módulos para aceleração para criptografia ou o que possuem é limitado. Esses módulos são importantes porque são hardwares que permitiriam aos microcontroladores criptografar dados que são gerados, armazenados ou transmitidos por eles e, assim, evitar invasões ou modificações dos dados por terceiros. Sem esse hardware ainda é possível utilizar algoritmos de criptografia, contudo pode ser ineficiente e exigir muita energia [3] o que pode inviabilizar o uso. Esses microcontroladores são usados como base para diferentes dispositivos IoT e a ausência de hardware para criptografia, além de poder de processamento normalmente limitado, torna necessário sistemas e protocolos que proporcionem segurança na comunicação dos dados gerados pelos dispositivos.

2.2 Blockchain

Blockchain pode ser definida um banco de dados descentralizado contendo blocos associados criptograficamente por transações assinadas digitalmente e governadas por um modelo de consenso. [4] A blockchain possui membros responsáveis por criar, propor e validar os dados e é descentralizada por não existir uma autoridade central ou alguma entidade específica responsável por fazer essas atividades. Todos os registros possuem marcadores de tempo (timestamp) e um resumo criptográfico único, de modo que o banco de dados

Microcontrolador	Clock Máximo (Mhz)	Aceleração para criptografia
ATmega328P	20	Não possui
ESP32	240	AES, SHA-2, RSA, ECC, RNG
MSP430FR6007	16	AES, RNG
STM32F103C8	72	Não possui
PIC16F877A	20	Não possui
Raspberry Pi 4*	1500	Não possui

Tabela 2.1: Microcontroladores e aceleração por hardware para criptografia
*Apesar de não ser um microcontrolador, o Raspberry Pi4 foi colocado na tabela pela sua alta popularidade e por ser uma plataforma de desenvolvimento muito usada em IoT.

possua um histórico auditável e imutável [5] de todas as transações na rede. Portanto, uma blockchain fornece acessibilidade, incorruptibilidade e a habilidade de armazenar e transferir dados de forma segura.

A arquitetura de uma blockchain consiste de blocos, nós, cadeia de blocos e algoritmos de consentimento.

Blocos

Os blocos correspondem à estrutura de dados que são armazenados no banco de dados. Cada bloco contém informações como, índice de ordenação de blocos, a data de criação do bloco, bytes de informação útil, duas palavras de verificação, que são chamadas de hash, uma que identifica unicamente o bloco atual e outra que identifica o bloco anterior.

O índice é definido como um valor incremental, a cada novo bloco criado, o índice corresponde ao do bloco anterior somado de um.

O hash do bloco é obtido selecionando algumas das informações do bloco ou todas, incluindo necessariamente o hash do bloco anterior. Essas informações são unidas, por exemplo, concatenando os caracteres, e depois é aplicado um algoritmo de hash, como o SHA256, para gerar um resumo criptográfico que seja difícil de ser obtido sem possuir as informações originais usadas para obtê-lo.

Como o hash necessariamente é criado usando o hash do bloco anterior, observa-se a importância da cadeia de blocos. Essa estrutura corresponde a uma sequência de blocos que estão conectados por meio do hash e estão ordenados pelo índice.

Algoritmo de Hash SHA-256

Os blocos da blockchain se relacionam por meio de hashes e, desse modo, funções que sejam capazes de gerá-los são necessárias. Um hash possui os objetivos de embaralhar dados de forma determinística, aceitar entradas de qualquer tamanho, produzindo uma

saída com um tamanho fixo e também o objetivo de gerar saídas como resultado que não possam recuperar a entrada, tornando o processo irreversível [6]. O SHA-256 é uma função hash que usa como entrada dados menores que 2^{64} bits e obtém uma saída com exatamente 256 bits [7]. Como a função depende dos bits dos dados de entrada, qualquer bit alterado gera um hash completamente diferente.

Nós

Os nós são os membros da rede da blockchain. Os nós podem ser usuários ou computadores e cada um deles possui uma cópia dos registros da blockchain. Os que realizam transações e geram dados para serem armazenados nos blocos são chamados de transacionais e os responsáveis por criar os blocos com esses dados são chamados de mineradores.

Algoritmos de Consenso

Os algoritmos de consenso são importantes por definirem as regras de como os nós da blockchain irão concordar sobre as transações que estão sendo geradas, de modo que todos eles possuam os mesmos blocos e possam criar novos deles que sejam confiáveis. Alguns algoritmos comumente usados são:

- Prova de Trabalho (PoW) - esse algoritmo se baseia na ideia de que, para um participante validar um bloco novo, ele deve realizar alguma forma de trabalho [8]. Esse trabalho deve ser difícil para quem o realizará, mas deve ser fácil para quem verificará que o trabalho foi realizado. Desse modo, o algoritmo normalmente exige que os nós que irão validar os blocos resolvam um problema matemático[9] e, em compensação, sejam recompensados de forma a incentivá-los a realizarem essa atividade. Essa recompensa somente é dada ao primeiro a encontrar essa solução e com isso ocorre um incentivo não somente a participar, mas também a possuir o maior poder computacional possível, já que, quanto maior, mais rapidamente é possível encontrar a solução [9, 10, 11]. Esse processo é conhecido como mineração e os nós responsáveis por realizá-lo são chamados de mineradores. Um problema matemático normalmente utilizado por esse algoritmo é o de encontrar um hash para o bloco que contenha uma quantidade de zeros no início definida pela dificuldade. Essa dificuldade é definida pelo algoritmo considerando o poder computacional dos mineradores de modo que cada novo bloco seja validado em um tempo semelhante. Como um byte diferente é suficiente para gerar um hash diferente, cada tentativa de um minerador consiste em concatenar um número diferente, chamado de nonce, aos dados do bloco e gerar um novo hash até encontrar um que cumpra a dificuldade exigida pelo algoritmo [8].

- Prova de Participação (PoS) - esse algoritmo surgiu como uma alternativa à prova de trabalho e tem o objetivo de permitir que um nó possa ser um validador sem precisar de um elevado poder computacional e, consequentemente, reduzir o elevado consumo energético causado pelos mineradores [9]. Nele, o processo de mineração é substituido por um processo de atestar os blocos (Attestation em [12]). Inicialmente um validador cria um novo bloco com os dados de uma transação e valida o bloco, por exemplo, criptografando o hash com uma chave privada e acrescentando o resultado como um novo atributo do bloco (o hash sem criptografia não é removido do bloco). Após a criação do bloco, é feito o processo de atestá-lo, no qual outros validadores verificam se não há problemas, sendo que no Ethereum 2.0 [12], existem 128 validadores por bloco. Os processos de criar o bloco e atestá-lo podem ser feito por qualquer participante da rede, contudo, a escolha de quem serão os validadores é feita de forma pseudo-aleatória com pesos. Esses pesos são definidos a partir da quantidade de tokens (moedas em sistemas de criptomoedas) que cada membro da rede separou para participar do sorteio, não sendo necessário utilizar todas os tokens que possui para isso [11, 13, 10]. Quando um nó é selecionado como validador, todos os tokens que ele utilizou no sorteio são bloqueados até o fim do processo de criar o bloco ou de atestá-lo. Após a finalização, todos os validadores participantes desses processos são recompensados com um bônus baseado na quantidade de tokens usados no sorteio. O consumo reduzido de energia em relação ao PoW pode ser observado pelo fato de não ser necessário que os validadores gerem uma enorme quantidade de hashes na adição de um novo bloco à blockchain, porém uma quantidade considerável de tokens é necessária para ter a chance de ser escolhido como um validador, em algumas blockchains existe, inclusive, um valor mínimo.

Processo de Verificação de Integridade dos Blocos

Um bloco é validado por um ou mais membros da rede que, no caso da prova de trabalho, é o primeiro a solucionar um problema matemático, no caso da prova de participação, são os escolhidos para validar o bloco. Os outros membros que não participaram dessa etapa devem confiar na validação gerada e isso é possível por os dados usados para validar o bloco estarem disponíveis para todos que possuem uma cópia da blockchain. Assim, quando um membro recebe um novo bloco para adicionar a sua cópia, no caso do PoW, ele pode facilmente verificar gerando um hash com o nonce obtido pelo minerador, a dificuldade da rede e as informações do bloco.

A Figura 2.1 mostra como as estruturas de uma blockchain são utilizadas para enviar uma transação quando o algoritmo de consenso é a Prova de Trabalho. Inicialmente um usuário (nesse caso Eduardo) que deseja realizar uma transação envia os bytes de

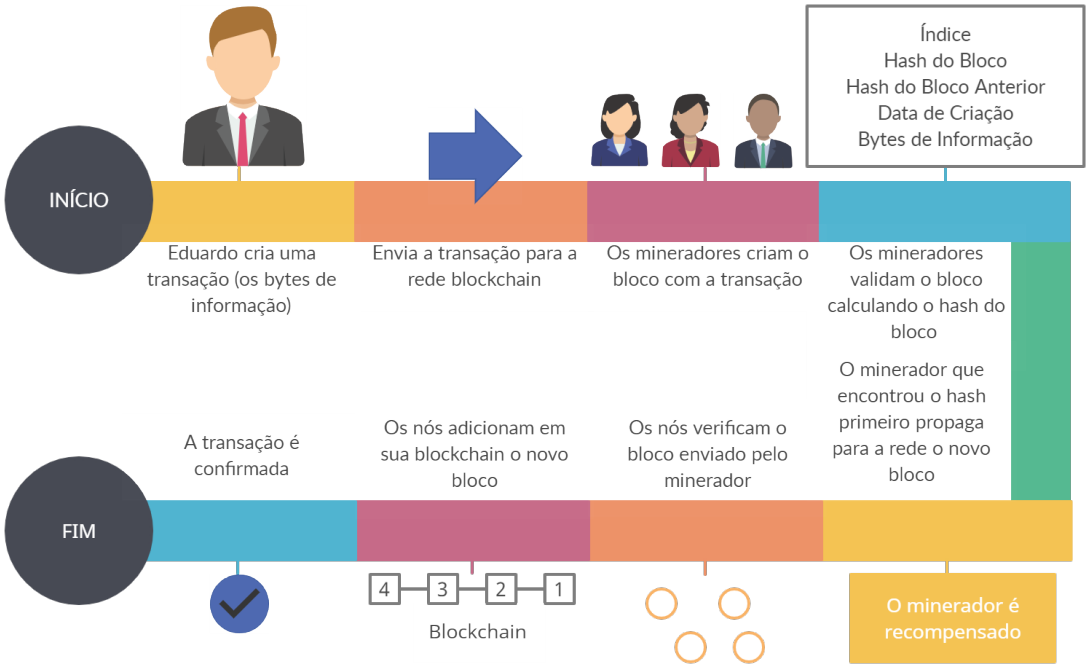

Figura 2.1: Processo para envio de Transação a uma Blockchain com PoW

informação para a rede contendo seu identificador único, os mineradores adicionam esses bytes ao próximo bloco. Esse bloco contém o índice, o hash do bloco anterior e a data de criação, além dos bytes de informação. Depois, cada minerador tenta encontrar o hash válido para o bloco. Quando um deles encontra o hash, adiciona-o no bloco, propaga o bloco para os outros nós da rede e recebe a recompensa pelo trabalho realizado. Em seguida, os nós que receberam o bloco verificam se o hash enviado está correto e se não existem problemas no bloco, caso não haja problema, o bloco é adicionado na blockchain de cada um desses nós e eles propagam para o restante da rede, confirmando a transação.

2.3 Docker

Docker é uma plataforma aberta para executar aplicações em ambientes isolados dentro de um mesmo computador que são chamados de containers. [14] Os containers são leves e cada um possui suas próprias configurações, com tudo que e necessário para executar a aplicação. Além disso, são facilmente compartilháveis, de modo que todas as pessoas que executarem uma cópia desse container irão conseguir executá-lo exatamente da mesma forma que quem o criou.

A Figura 2.2 mostra a arquitetura do Docker, que, como pode ser observado, é formada por:

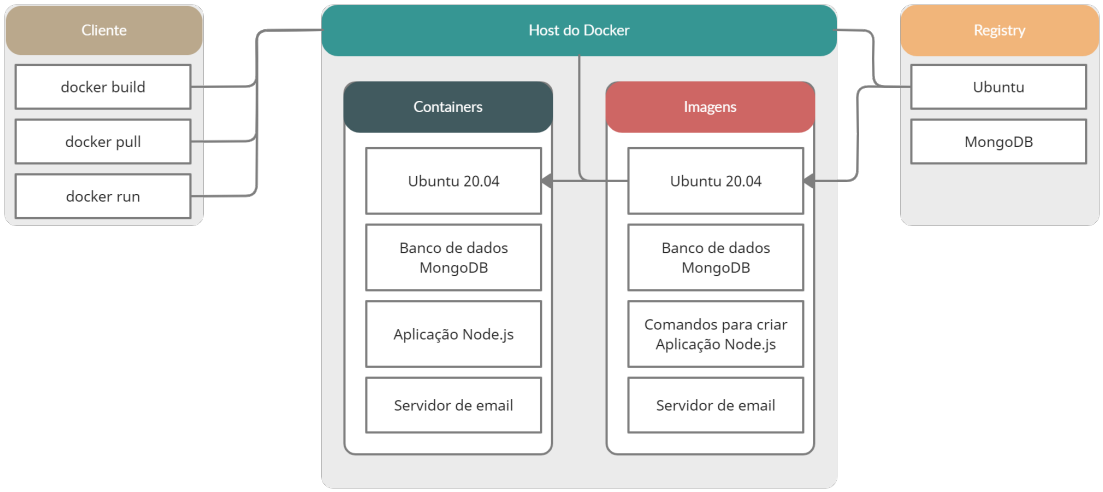

Figura 2.2: Arquitetura do Docker

- Host do Docker - é um computador ou um servidor com Windows, Linux ou Mac OS X que executa o Docker daemon e serve de anfitrião para os containers que são executados no Docker;
- Docker daemon - é responsável por gerenciar os objetos do docker e observar requisições à API do Docker;
- Objetos do Docker - são os containers, as imagens, os volumes e as redes além de outros objetos que existem na plataforma. As imagens são templates com as instruções para criar um container. Os containers, responsáveis por executar a aplicação, são instâncias de uma imagem que podem se conectar a uma ou mais redes, possuir armazenamento ou servir como base para gerar uma nova imagem. Volumes são mecanismos para persistência de dados de containers. Já as redes funcionam de modo que um container possa se comunicar com outros ou ainda possam se comunicar com serviços que não estão no Docker, mas sim no sistema anfitrião. Em 2.2, é possível observar dois tipos de objetos, os containers e as imagens, a imagem, por exemplo, do Ubuntu 20.04 diz o que o Docker deve fazer para conseguir inicializar um container que executa o Ubuntu 20.04. Após a inicialização do container, o usuário pode criar, instalar ou executar qualquer arquivo ou programa que funcione com o Ubuntu 20.04 como se o computador dele fosse esse sistema e tudo que é feito dentro do container não interfere no funcionamento do sistema operacional anfitrião do usuário. Além disso diferentes aplicações podem ser executadas como um container, um banco de dados, um servidor de email, um cliente de torrent, um serviço de proxy e cada um executa sem interferência de outro container;

- Cliente do Docker - é a forma como os usuários podem interagir com a plataforma. Existem comandos usados pelo usuário que o cliente envia para o daemon executar;
- Docker Registry - um registry é um repositório que armazena imagens do Docker, um exemplo é o Docker Hub, que é um repositório público que os usuários podem usar para enviar imagens para outros utilizarem. Algumas imagens que podem ser obtidas no Docker Hub são o Ubuntu e o MongoDB em diferentes versões.

O Docker possui alguns comandos que são importantes para executar os containers:

- docker run - inicializa um container a partir de uma imagem, inicialmente procura a imagem localmente no sistema anfitrião e, se não possuir, procura em um repositório;
- docker pull - procura uma imagem em um repositório, se já existe uma imagem local essa imagem é atualizada para uma nova versão se existir;
- docker build - cria uma imagem a partir de um arquivo com nome Dockerfile.

Além desses comandos, o Docker ainda possui uma ferramenta para definir e inicializar vários containers de uma única vez, já que somente com o comando seria necessário inicializar um por um. Essa ferramenta é o Docker Compose que permite configurar os containers que serão inicializados, as configurações de rede e os volumes que serão utilizados em um arquivo no formato YAML e de nome docker-compose.yml. Todas as configurações colocadas nesse arquivo podem ser inicializadas com docker-compose up, se houver uma alteração no arquivo esse mesmo comando também consegue recriar um container ou alguma configuração que tenha sido alterada. Além desse comando, também há o comando docker-compose pull que baixa ou atualiza as imagens que vão ser utilizadas pelos containers e também há o comando docker-compose build que cria imagens a partir de arquivos identificados no docker-compose.yml.

2.4 Ataque Man-in-the-middle

O ataque Man-in-the-middle (MITM) é uma forma de invasão antiga, existente até mesmo antes do surgimento dos computadores [15], e que, até hoje, continua sendo utilizado, inclusive em dispositivos IoT [16]. Esse ataque consiste em interceptar a comunicação entre 2 pontos e pode ter diferentes objetivos a depender do alvo. Esses objetivos podem ser obter os dados enviados nas mensagens, alterar o conteúdo delas, atrasá-las ou ainda excluí-las.

A Figura 2.3 mostra como fica a conexão entre 2 usuários quando um man-in-the-middle invade a conexão, a comunicação que antes era feita diretamente entre os dois,

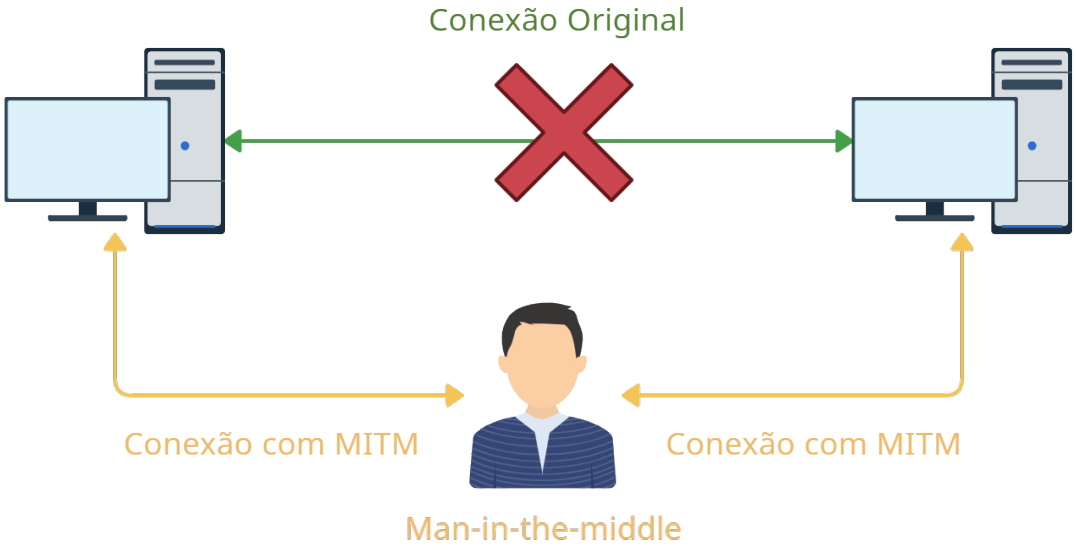

Figura 2.3: Conexão entre 2 usuários com Man-in-the-middle

representada pela linha verde, é bloqueada e ela passa a ser feita do usuário da esquerda (A) para o man-in-the-middle e do man-in-the-middle para o usuário da direita (B), representada pelas linhas amarelas. Já na Figura 2.4 é exemplificado o que um man-in-the-middle pode fazer ao interceptar a comunicação. No primeiro caso, o usuário A envia `Olá` e o man-in-the-middle não faz nada e o usuário B recebe `Olá`. No segundo caso, o usuário B envia um endereço para pagamento `123456` e o man-in-the-middle altera essa mensagem enviando para o usuário A o endereço `456789` e, assim, o usuário A faz o pagamento para a pessoa errada. No último caso, o usuário A solicita o endereço do usuário B, mas o man-in-the-middle impede que o usuário B receba essa mensagem.

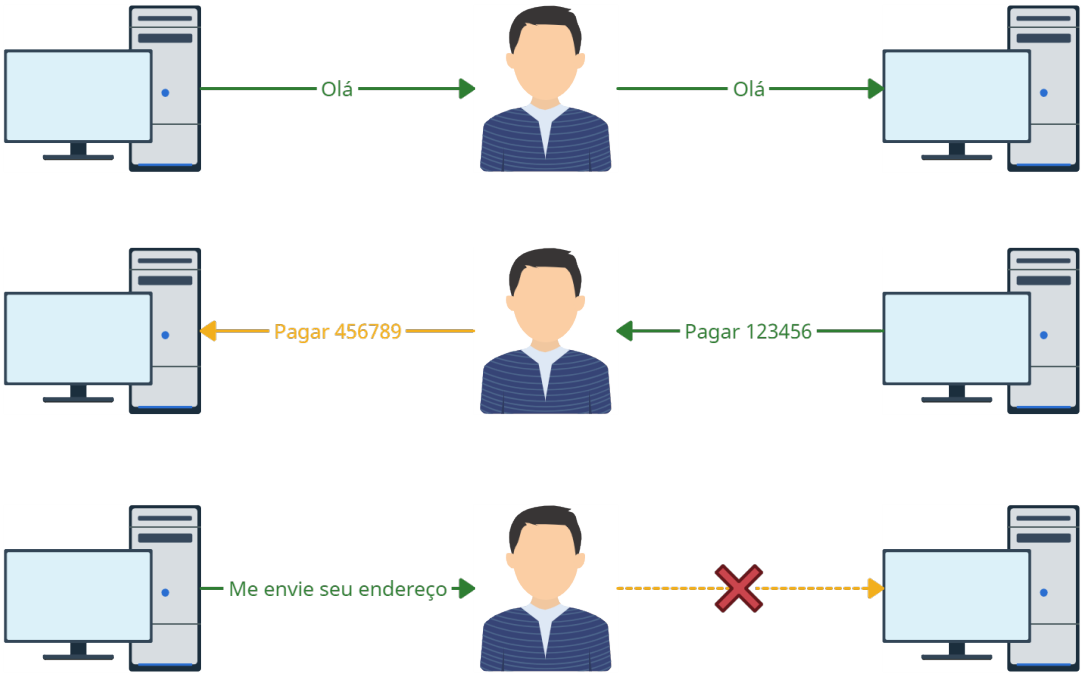

Figura 2.4: Mensagens trocadas entre 2 usuários com Man-in-the-middle

Capítulo 3

Metodologia

O modelo do protocolo foi feito usando uma blockchain e a estrutura da Figura 3.1 e com os processos definidos na Figura 3.2.

A estrutura consiste de:

- Dispositivos - são responsáveis por coletar dados de equipamentos.
- Nós mineradores - são responsáveis por criar blocos com hashes válidos.
- Nós centrais (ou hubs) - são responsáveis por armazenar em um banco de dados os blocos criados pelos mineradores.
- Atores - são pessoas externas que acessam a blockchain armazenada nos nós centrais.

Essa estrutura considera que existe uma blockchain privada, construída em uma rede interna e que apenas a blockchain com os blocos já validados é fornecida para a rede externa. Com isso, determinados tipos de ataque como ataque dos 51% ou ataque Sybil podem ser prevenidos, já que todos os nós responsáveis por criar e armazenar os blocos são considerados confiáveis e trabalham de maneira justa. Além disso, outros ataques que poderiam acontecer entre dispositivos, nós mineradores e nós centrais, que são as estruturas que participam da coleta de dados e criação da blockchain, não são o foco do protocolo. Assim, esse protocolo foi projetado visando impedir ataques man-in-the-middle que poderiam acontecer entre o responsável por entregar os dados (nós centrais) e as pessoas externas que precisam usar os dados (atores).

Ataques que não ocorrem por depender de nós maliciosos entre os mineradores:

- Ataque Sybil - Esse ataque visa assumir o controle da rede toda, para tentar direcionar as transações. Para isso, é necessário possuir a maioria dos nós conectados na rede.

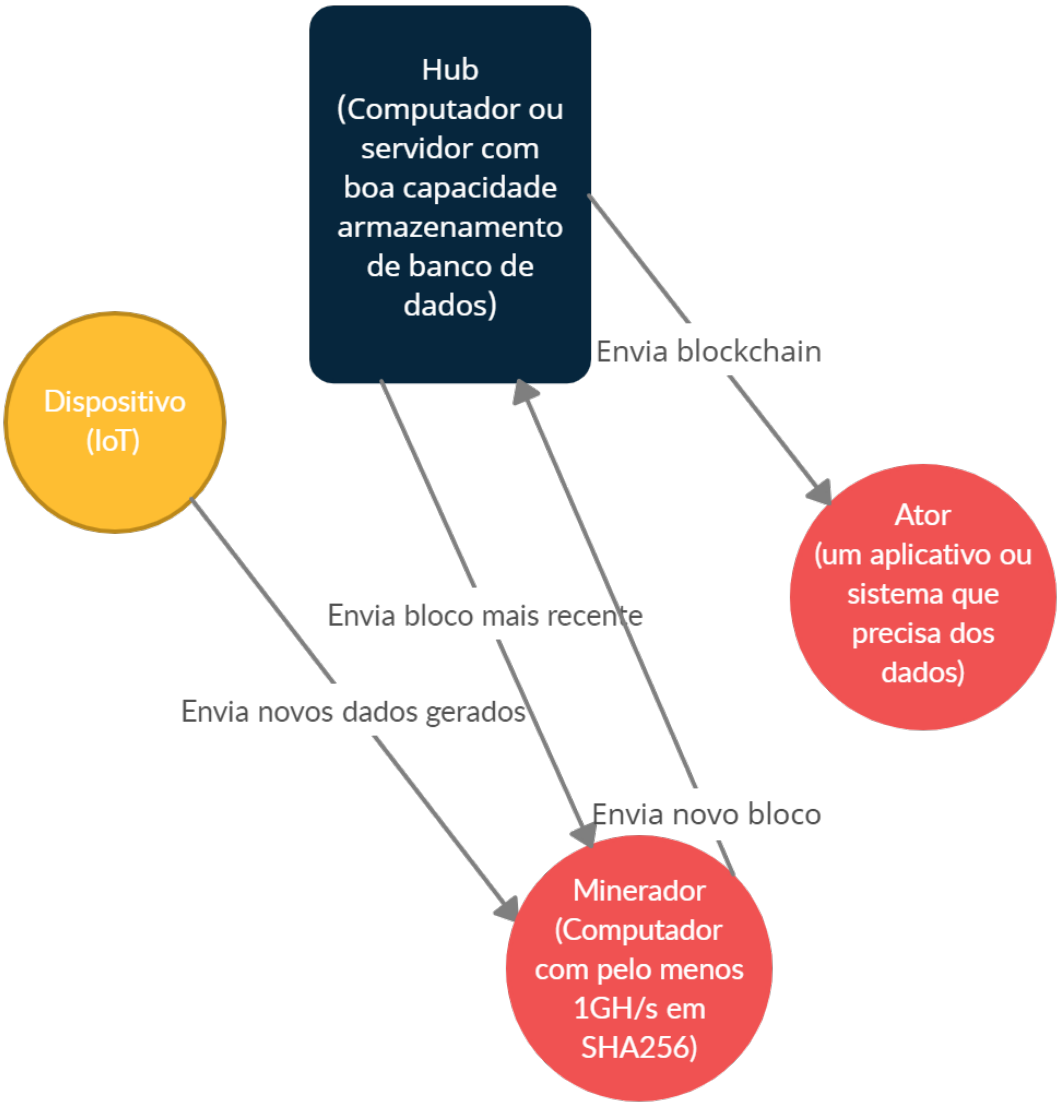

Figura 3.1: Estruturas presentes na blockchain

- Ataque dos 51% - Esse ataque, em vez de ter a maioria dos nós como no Ataque Sybil, possui o objetivo de possuir o maior poder computacional, no caso, correspondente a 51% do poder computacional da rede.

Na Figura 3.1 são apresentadas as estruturas do protocolo mostrando as funções que cada dispositivo e de forma simplificada as atividades desempenhadas por cada um. O papel de dispositivos é formado por dispositivos IoT que coletam dados e os enviam para os mineradores. Os mineradores são computadores com capacidade de pelo menos 1 gigahash por segundo para gerar hashes SHA256. Esses mineradores recebem dados dos dispositivos e solicitam o bloco mais recente da blockchain aos hubs para que possam criar um novo bloco. Após criar esse novo bloco, os mineradores enviam para os hubs.

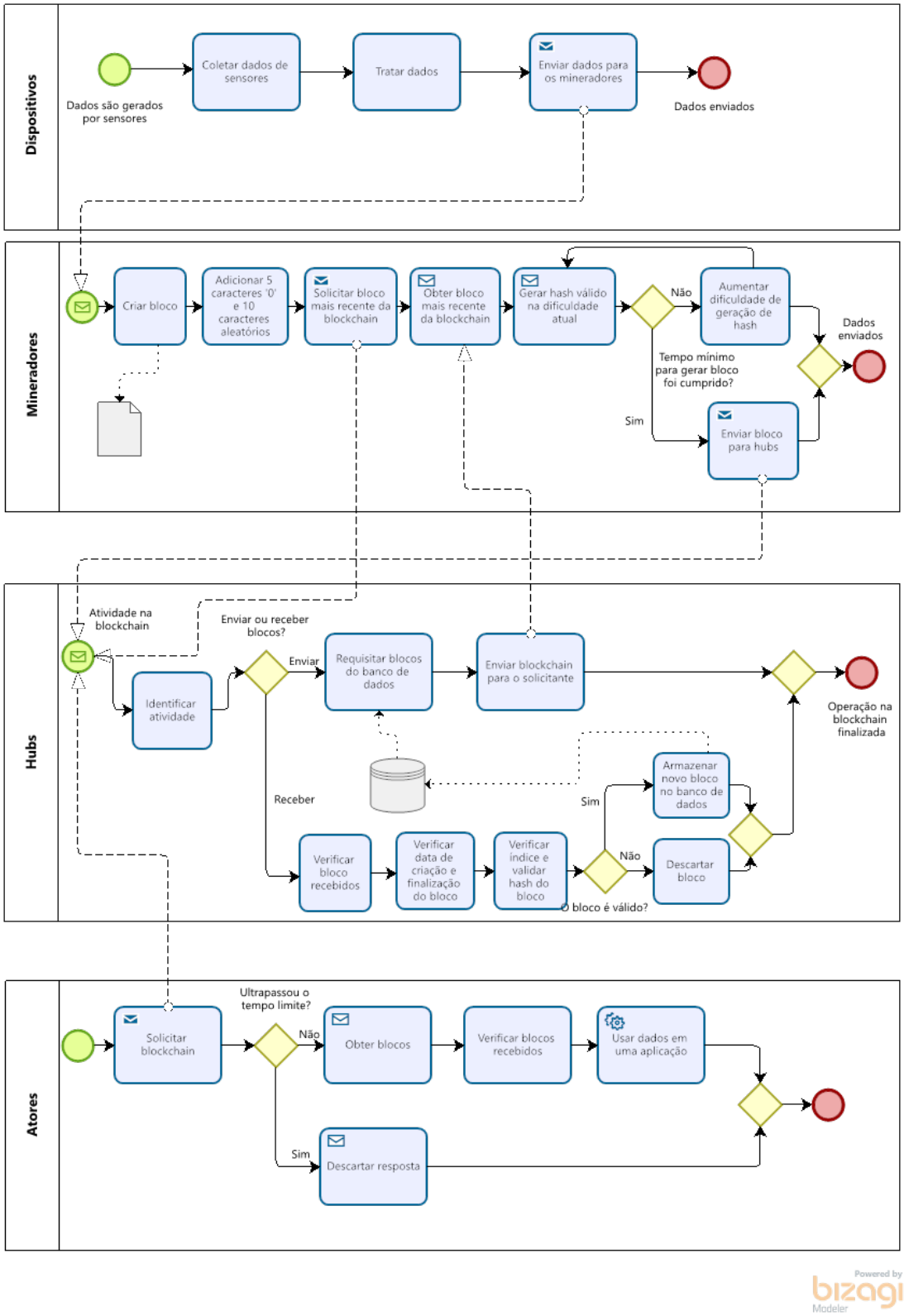

Figura 3.2: Processo de coleta e envio de dados com o protocolo de blockchain

Os hubs são computadores ou servidores com grande capacidade de armazenamento para guardar os blocos em banco de dados. Esses hubs recebem novos blocos para armazenar e enviam o bloco mais recente para mineradores, além de enviarem a blockchain quando solicitado por um ator. Os atores são aplicativos ou sistemas que utilizam os dados da blockchain.

Na Figura 3.2 são descritas as etapas do processo desde a criação de um bloco até a requisição da blockchain contendo o bloco. Inicialmente os dispositivos coletam dados de equipamentos e máquinas, na sequência tratam esses dados e enviam como uma transação para os mineradores. Isso encerra a participação dos dispositivos na criação de um bloco. Após isso, os mineradores coletam os dados enviados pelos dispositivos, criam um bloco e acrescentam cinco '0' e 10 caracteres aleatórios nos bytes de informação. Depois solicitam para os hubs (ou nós centrais) a blockchain contendo o bloco mais recente para acrescentarem o hash do último bloco que foi validado como sendo o hash bloco anterior ao que estão criando. Com todos os dados prontos, os mineradores iniciam de fato o processo de mineração gerando o hash válido do bloco. Caso o tempo mínimo para validar o bloco não tenha sido cumprido, a dificuldade é aumentada e um novo hash válido deve ser obtida, essa etapa é repetida até obter um hash válido com tempo maior ou igual ao mínimo. Quando isso é cumprido, o novo bloco é enviado a todos os hubs para que possam armazená-lo no banco de dados. No hub, existem 2 possibilidades enviar ou receber blocos, quando os mineradores solicitam o bloco mais recente, é a atividade de enviar, no qual cada hub obtém do banco de dados a blockchain e envia para os mineradores. Além disso, a solicitação pode ser de um ator e é feito o mesmo, obtém-se a blockchain do banco de dados e envia para o ator. Quando a atividade é de recebimento, um bloco novo é recebido de um minerador. Esse bloco é verificado observando se a data de criação e finalização do bloco cumprem o tempo mínimo, se o índice é o seguinte ao mais recente armazenado na blockchain, e se o hash obtido pelo minerador corresponde ao do bloco. Se não houver nenhum problema na verificação o novo bloco é armazenado na blockchain, caso contrário é descartado. Os atores possuem a função de utilizar a blockchain em algum sistema ou aplicativo, logo, para o processo, a atividade deles é somente de solicitar a blockchain aos hubs. Ao receber a blockchain, para garantir que não houve modificação na blockchain durante o envio dos hubs, os atores verificam se o tempo até receberem não ultrapassou um tempo limite definido por eles mesmos antes de requisitarem a blockchain, caso ultrapasse os blocos recebidos são descartadas pela blockchain, caso contrário verificam se os blocos são válidos e depois usam nos sistemas ou aplicativos deles.

3.1 Blockchain

Blocos

Os blocos da blockchain possuem hash, hash do bloco anterior, bytes de informação útil, índice, data de criação, data de atualização, dificuldade e nonce. Os bytes de informação possuem um tamanho máximo definido na inicialização da blockchain e inalterável. Quando os dados adicionados a um bloco são menores que o tamanho máximo, são acrescentados caracteres '0' até completar o tamanho máximo. Após ser garantido que os dados possuem exatamente o tamanho máximo, são adicionados 5 caracteres '0' e depois mais 10 caracteres aleatórios ao final dos dados.

Algoritmo de Consenso

O algoritmo de consenso da blockchain é a prova de trabalho. Desse modo, os nós mineradores precisam encontrar um hash válido alterando o nonce em um tempo mínimo de 24 segundos. Inicialmente, a dificuldade é definida como 0 (não é necessário nenhum 0 no início do hash). Caso o tempo seja inferior ao mínimo, a dificuldade deve ser incrementada em 1, exigindo que o novo hash válido tenha um 1 zero a mais no início dele. Assim, o bloco torna-se válido quando um hash possui a quantidade de zeros no início igual à dificuldade e o tempo necessário para obtê-lo é superior ao tempo mínimo.

3.2 Dispositivos

Os dispositivos iniciam o processo coletando os dados de equipamentos e máquinas com sensores, fazem um processamento acerca dos dados que estão sendo gerados e, em seguida, enviam para os nós mineradores. Em uma blockchain, corresponde aos usuários que desejam criar uma nova transação.

Normalmente esses dispositivos como evidenciado na Figura 3.1 são microcontroladores para IoT, já que não precisam de considerável poder computacional, mas precisam de sensores ou outros recursos para uma aplicação específica e se comunicam pela internet com os outros membros da blockchain.

3.3 Nós mineradores

Os nós mineradores por serem responsáveis por criar os blocos devem possuir poder computacional suficiente para gerar muitos hashes com SHA-256 a cada segundo, como evidenciado na Figura 3.1 em que é indicado um computador com pelo menos 1 Gigahash por

segundo (GH/s). Normalmente computadores que possuem unidades de processamento gráfico dedicadas. Isso é necessário para que os blocos possuam hashes de elevada dificuldade e consequentemente exijam um tempo significativo de invasores ou interceptadores de dados que tentem modificar a blockchain.

Quando os nós mineradores são notificados sobre novos dados enviados pelos dispositivos, cada um deles cria um novo bloco com os mesmos dados e garantem que os bytes de informação possuam exatamente o tamanho definido e tenham os 5 '0' e os 10 caracteres aleatórios. Em seguida, solicitam para os nós centrais uma cópia da blockchain para identificarem o último bloco gerado, já que o índice do último bloco e o hash do bloco anterior são informações necessárias para a criação de um bloco válido. Obtidas essas informações, os mineradores iniciam o processo de geração de um hash válido usando um dos algoritmos de consenso.

Após a criação do hash, o bloco é enviado para todos os nós centrais.

3.4 Nós centrais

Os nós centrais por armazenarem a blockchain precisam de grande espaço de armazenamento como identificado na Figura 3.1 para que possam guardar todos os blocos criados, já que um único bloco pode ter 1MB e quanto maior a blockchain mais segura ela é contra ataques que tenham o objetivo de modificar a blockchain.

Esses nós recebem simultaneamente o bloco gerado por um mesmo minerador e verificam se o bloco criado é realmente válido. Caso seja válido, esses nós armazenam o novo bloco no banco de dados que cada um deles possui.

O processo de verificação é necessário, já que, como os mineradores estão tentando gerar um bloco com os mesmos dados ao mesmo tempo, pode ocorrer de um minerador "A" ter enviado um bloco de mesmo índice de um já validado, que foi enviado por um minerador "B" antes. Outro fator para essa verificação ocorrer é que pode ter acontecido uma assincronia entre a blockchain dos nós centrais por diferentes motivos, como, por exemplo um desses nós estava desligado quando novos blocos estavam sendo gerados.

Além de serem responsáveis por armazenar uma cópia da blockchain em um banco de dados, os nós centrais também são responsáveis por realizar a sincronização entre si. Essa sincronização pode ocorrer na inicialização de um novo nó central na rede, pode ocorrer periodicamente ou quando detectada uma falha na inserção de um novo bloco, geralmente por uma diferença significativa entre o índice do bloco mais recente no banco de dados e o índice do recebido por um minerador.

Outra função desses nós é enviar uma cópia da blockchain para os nós mineradores. Todos os nós tentam enviar simultaneamente e os mineradores escolhem a primeira cópia recebida para usar no processo de criar um novo bloco.

A última atividade realizada por esses nós é a de enviar uma cópia da blockchain para os usuários externos. Assim, esses são únicos nós que precisam ter acesso à internet, já que os usuários externos não necessariamente estão na rede local. O envio desses dados pode ser complementado por outros protocolos para obter outro tipo de segurança dos dados, como a privacidade, fator que não é garantido somente pela blockchain.

3.5 Atores

Os atores são aplicações que podem ou não estar estar na rede local e que precisam dos dados gerados nos dispositivos IoT, como notado pela Figura 3.1. Pode ser de interesse o compartilhamento dos dados gerados pelos dispositivos com instituições, empresas ou pessoas que não estão relacionadas diretamente com os responsáveis pela blockchain. Esse é o principal objetivo do protocolo, poder comunicar os dados com terceiros de forma que os dados sejam invioláveis para que esses usuários tenham a garantia de que as informações que estão recebendo não sofreram nenhuma modificação que as torne inválidas.

Os atores, quando desejam receber alguma informação dos dispositivos, solicitam a um dos nós centrais uma cópia da blockchain e, para garantir que estão recebendo corretamente os dados, o aplicativo desses usuários observa quanto tempo demorou para receber uma resposta à requisição, caso o tempo seja superior a um valor definido pelo próprio aplicativo no momento da requisição, a resposta é descartada pelos usuários e deve ser feita uma nova solicitação ou deve ser verificado se existe algum problema na comunicação entre o nó e a aplicação.

O uso de um intervalo de tempo como sendo a garantia de ter recebido dados válidos ocorre por ser necessário um tempo elevado para que intermediários tentem modificar a blockchain. Como o hash do bloco atual depende do hash do anterior, é necessário refazer todos os blocos da blockchain a partir do modificado de modo a obter uma blockchain válida. Desse modo, se existem 100 blocos e o bloco 10 é alterado, são necessários reencontrar o hash de 91 blocos. Isso permite obter uma equação que indique quanto tempo é esperado para validar uma blockchain quando um bloco de índice n é alterado e considerando que a quantidade de hashes H necessária para um bloco ser validado não se altera quando o bloco é modificado. A equação 3.1 relaciona ainda o tempo T necessário para obter um bloco válido e a quantidade de hashes por segundo H_t que um dispositivo é capaz de gerar. Além disso, com uma quantidade N suficientemente grande de blocos e muito maior que n, é possível considerar o tempo para validar cada bloco como sendo o

mesmo e igual ao tempo médio e o tempo total T_{total} para alterar a blockchain pode ser aproximado para a equação 3.2.

Pela equação 3.2, é possível observar que no caso de o invasor possuir o mesmo poder computacional dos mineradores, para alterar o bloco 10 em uma blockchain com 100 blocos e tempo médio de 30 segundos, seriam necessários 45 minutos para alterar a blockchain e tornar os novos valores válidos. Além disso, se considerado que uma requisição de uma aplicação de um ator para um hub não deve demorar mais de 30 segundos, o ator descartaria a blockchain recebida do atacante. Portanto, a partir disso, é possível perceber a necessidade de o invasor possuir 90 vezes o poder computacional dos mineradores.

$$T = H/H_t \tag{3.1}$$

$$\begin{aligned} &\text{Para um N} \gg \text{n:} \\ &T_{total} = \text{N} \times H/H_t \end{aligned} \tag{3.2}$$

3.6 Ambiente de testes

O protocolo foi testado em um único computador com as configurações:

- Processador AMD Ryzen 7 2700
- Memória DDR4 32GB 3200Mhz
- SSD 240GB
- Sistema Operacional Manjaro 20.2
- Placa de vídeo Nvidia GTX 1060 6GB*

* Os testes não utilizaram a placa de vídeo apesar de o processador gerar menos hashes por segundo. Isso ocorreu para simplificar os testes e pelo fato de que o man-in-the-middle possuía exatamente o mesmo desempenho dos mineradores em gerar hashes. A recomendação para uso de placas de vídeo em mineradores é devido ao desempenho superior em relação a processadores e por serem acessíveis ao consumidor final e por não possuírem preço elevado se comparado a outras soluções existentes no mercado.

Para aplicar o modelo explicado no capítulo anterior, foi construída uma blockchain usando *Node.js*, *MongoDB* e *Socket.io*.

O banco de dados utilizado para armazenar os blocos da blockchain foi o *MongoDB*.

O *Node.js* foi usado para a definição de rotas para requisições HTTP de criação de novas transações pelos dispositivos e também de solicitação da blockchain armazenada

no banco de dados pelos usuários. Além disso, também foi usado para a definição da estrutura dos dados guardados no banco de dados e para a comunicação por socket usando o *Socket.io*.

O *Socket.io* foi usado para fazer a comunicação entre os nós mineradores e os nós centrais por comunicação peer-to-peer usando socket e eventos.

Implementação das estruturas

As estruturas foram separadas em containers do Docker, de modo a simular o funcionamento da blockchain em múltiplos pares com configurações e ambientes distintos. Cada container corresponde a um nó minerador ou a um nó central, podendo ter vários nós desses tipos, que se comunicam por socket.

Além desses containers, os dispositivos possuem um nó separado responsável por receber as requisições e enviar os dados para os nós conectados na rede.

Os usuários podem se comunicar diretamente com cada nó central e solicitar a blockchain armazenada em cada um deles.

A Figura 3.3 mostra como estavam configurados os containers do Docker para o ambiente de testes. Um computador que foi utilizado como host para o Docker possuía um total de 8 containers, 2 eram mineradores (em verde) que eram aplicações em Node.js que desempenhavam a função de receber dados de dispositivos, criar novos blocos e validar os dados, incluindo um hash válido, além de enviar os novos blocos para os hubs. Outros 3 containers eram hubs (em azul) e cada um deles possuía um outro container associado (em amarelo) que correspondia a um banco de dados MongoDB. Esses hubs possuíam a função de armazenar novos blocos no banco de dados e enviar a blockchain para os mineradores se solicitado ou a um ator.

Simulação de Ataque Man-in-the-middle

Para simular uma invasão de man-in-the-middle, um container malicioso com o mesmo desempenho dos mineradores foi criado, no qual é responsável por ser um intermediário entre um ator e um hub. Esse container obtém a blockchain quando solicitado por uma aplicação e tenta modificar os dados com o objetivo de enganar a aplicação fazendo com que ela realize uma atividade diferente do esperado. Nessa simulação, o ator, por possuir um limite de tempo, deve verificar a presença de um potencial ataque.

A Figura 3.4 mostra como os containers interagem entre si e também com os atores (aplicativos) e dispositivos. Além disso, mostra ainda como um aplicativo se comunica com um hub quando um MITM é introduzido. A comunicação entre mineradores e hubs é feita por socket e eventos utilizando o Socket.io, tanto para receber, quanto para enviar

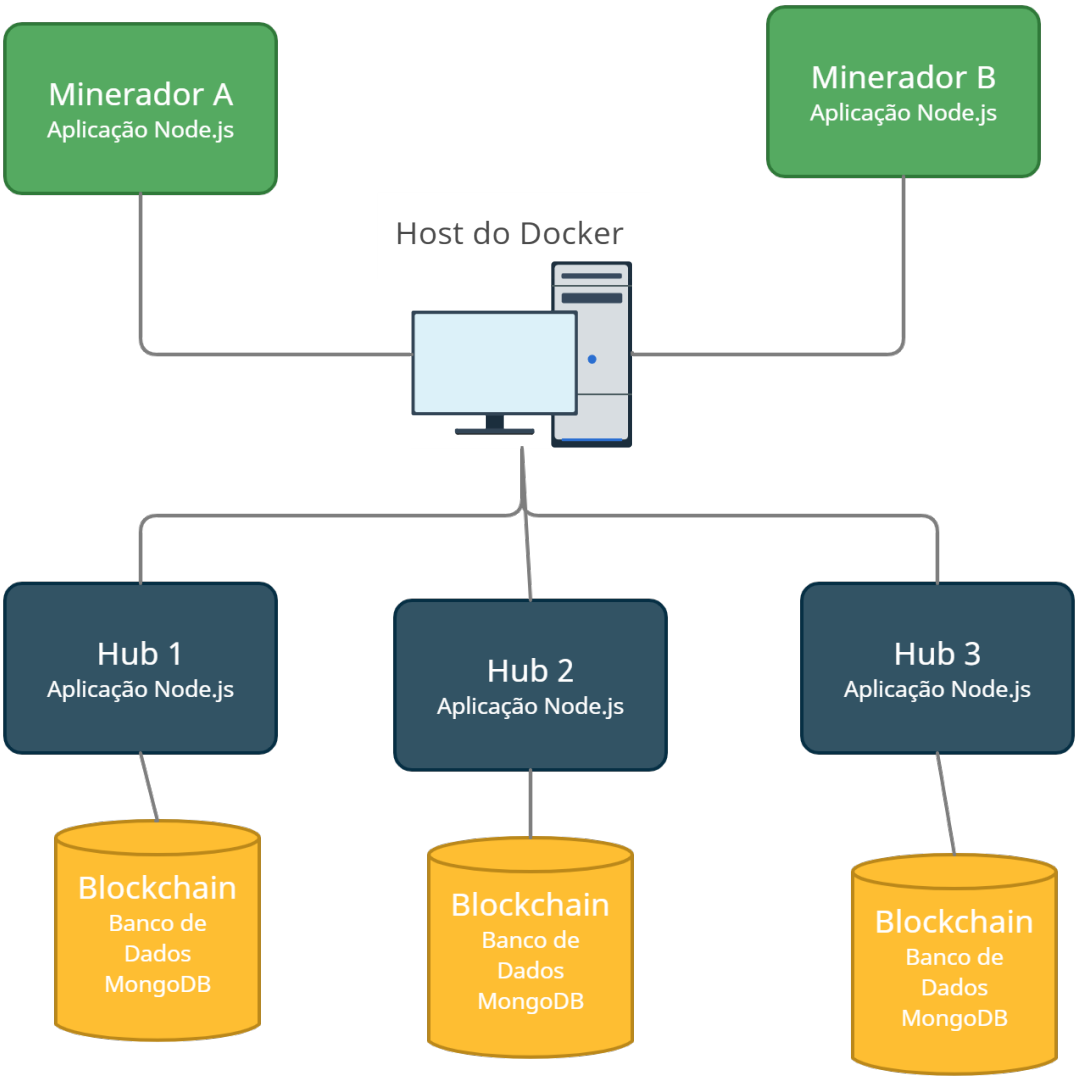

Figura 3.3: Containers criados para o ambiente de testes

blocos, a comunicação de um dispositivo com os mineradores também é feita da mesma forma, ele envia os dados por meio de socket. Já a conexão entre um hub e um aplicativo é feita por requisições HTTP, o aplicativo solicita a blockchain ao hub e o hub responde enviando a blockchain. O MITM é uma aplicação Node.js em um container capaz de ler uma blockchain, alterá-la e revalidar os blocos, que fica entre um hub e um aplicativo. Quando um aplicativo que teve a conexão invadida solicita a blockchain a um hub, na verdade ele solicita ao MITM e o MITM solicita ao hub. Na sequência, o hub responde ao MITM enviando a blockchain e o MITM altera alguns blocos, revalida-os e entrega ao aplicativo a blockchain modificada. O aplicativo A da Figura 3.4 mostra como uma conexão seria feita entre ele e um hub e o aplicativo B como essa mesma conexão é feita quando há um MITM.

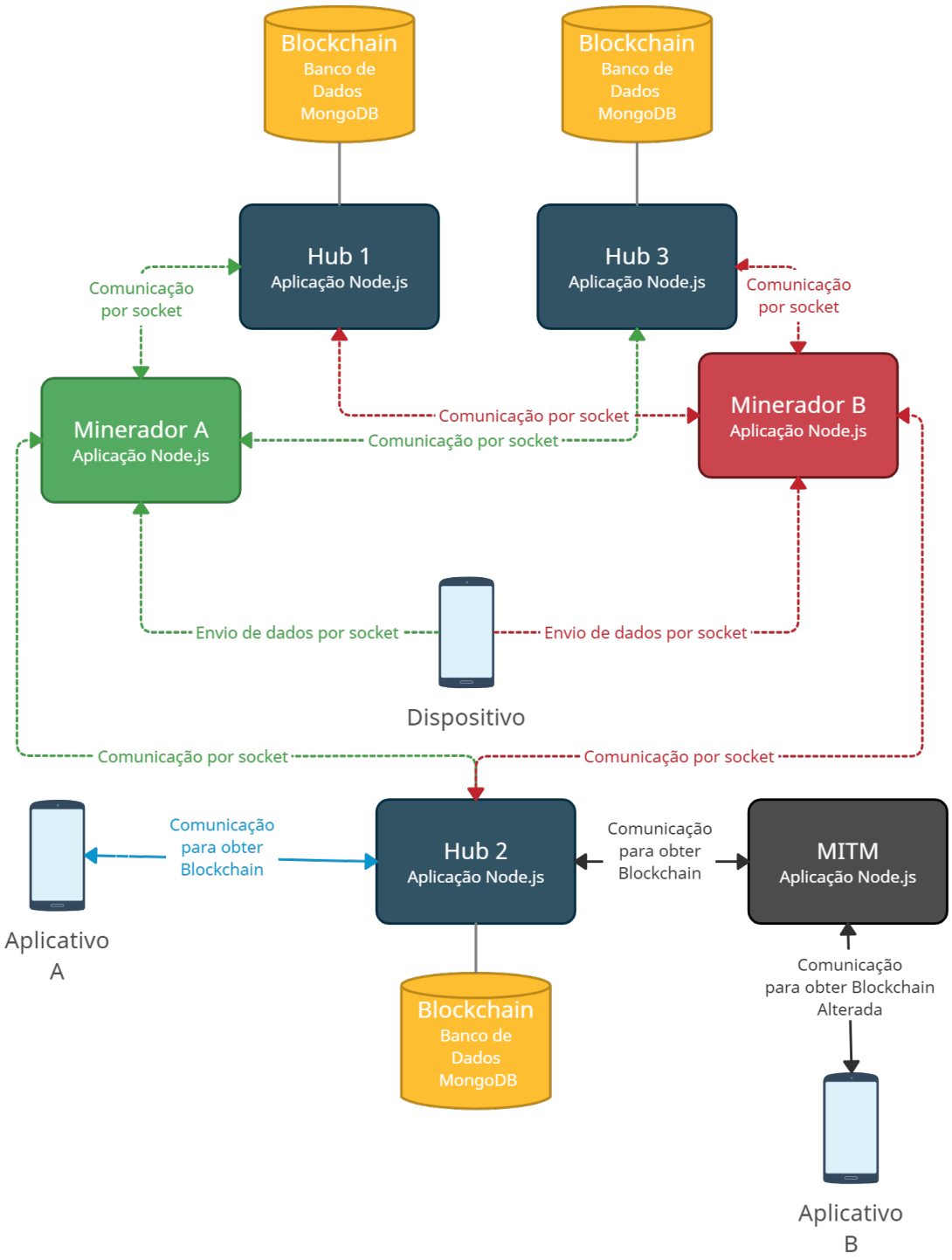

Figura 3.4: Ambiente de testes com introdução do MITM, dispositivos e aplicativos

Capítulo 4

Resultados

Estudo de Caso

Foi feito um estudo de caso, demonstrando como funciona o protocolo na geração de um bloco. Considerou-se que existia apenas um bloco já existente na blockchain, o bloco chamado de gênesis, que é auto-gerado antes de inicializar o protocolo. No caso, o objetivo é verificar o que acontece com uma palavra `teste` recebida por um dispositivo e que deve ser enviada para uma aplicação de terceiros.

Como observa-se na figura 4.1, o dispositivo envia a palavra `teste` para os mineradores (nesse caso só existia um minerador).

Após o minerador receber a palavra, ele acrescenta a data de criação do bloco, o índice do novo bloco e o hash do bloco anterior e inicializa a dificuldade para a geração do hash como 0. Em seguida, ele começa o processo de criação do hash com nonce 0 e vai aumentando até encontrar um hash válido para a dificuldade atual. Depois verifica, se cumpriu o tempo mínimo para geração do hash, se não cumpriu aumenta a dificuldade e retoma o processo de geração de hashes. Quando o minerador alcança a dificuldade 14, o tempo já foi cumprido e o hash válido para essa dificuldade é obtido como sendo `0003a33ad2ad06381e68b269a09c9d395378034eff6733313eeb5b5308192f23`.

Com o bloco formado e validado pelo minerador, ele envia para os hubs (nesse caso existia apenas um), que é responsável por manter a blockchain armazenada em um banco de dados. Posteriormente, a aplicação que deseja obter os dados solicita a blockchain para o hub e a recebe.

Para esse estudo de caso, seguindo as estruturas apresentadas na figura 3.1 seriam necessários um microcontrolador, como por exemplo um ESP32, para gerar os dados e fazer o papel de dispositivo da da estrutura da blockchain; um computador com disco rígido grande o suficiente para armazenar a blockchain e uma placa de vídeo para fazer o

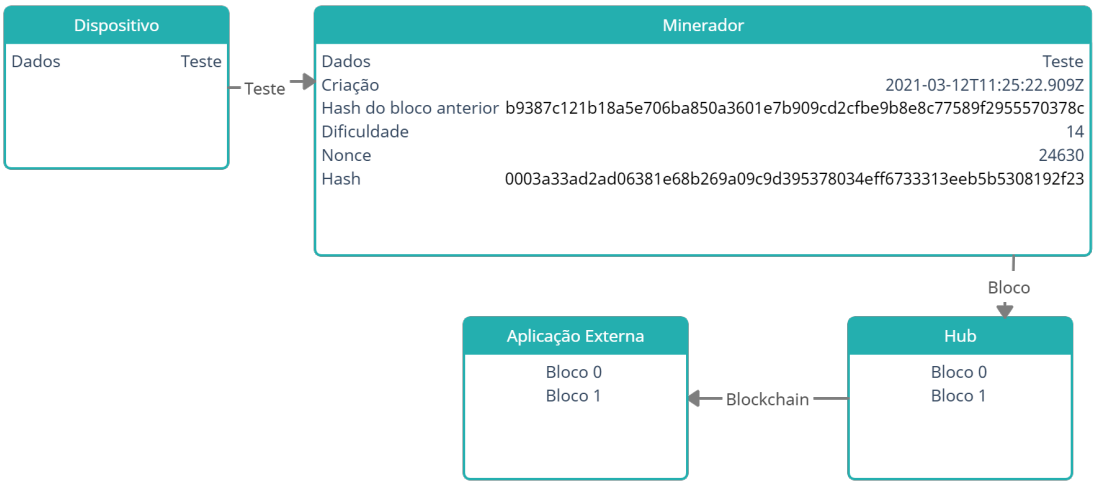

Figura 4.1: Processo de coleta e envio de dados com o protocolo de blockchain

papel de nó minerador e hub; e um servidor contendo um sistema para utilizar os dados da blockchain e fazer o papel de ator.

Testes

Os testes foram separados em 2 etapas. Na primeira delas, o objetivo era prever qual seria o tempo esperado para que um man-in-the-middle conseguisse alterar uma blockchain contendo de 1 a 10 blocos considerando que o poder computacional dele era o mesmo dos mineradores. Nesse teste, um usuário solicitava a um nó hub a blockchain e um nó modificado tentava alterar algumas informações dos bytes de informação de cada bloco e obtinha uma blockchain válida com a mesma dificuldade dos blocos originais e posteriormente enviava para o usuário que requisitou os dados. Pela equação 3.2, se a quantidade de hashes necessária para validar a blockchain permanecesse inalterada, o tempo necessário para o nó modificado entregar ao usuário os dados seria n vezes o tempo para obter um bloco, sendo n a quantidade de blocos. No teste, o tempo mínimo para o bloco ser gerado era de 24 segundos (o tempo alvo era de 30 segundos, mas era permitido que blocos fossem gerados com tempo 20% inferior ao alvo) e, assim, a expectativa era de que os tempos fossem próximos de 24 segundos para 1 bloco e 240 segundos para 10 blocos. Entretanto, após 8 repetições do teste, o tempo médio para modificar os blocos foi superior à expectativa, como pode ser observado na tabela 4.1. Além disso, foi obtido a Figura 4.2 indicando a curva do tempo esperado para modificar a blockchain em relação ao número de blocos. Nesse gráfico, é possível perceber que o tempo previsto corresponde a uma curva linear o que coincide com a Equação 3.2, já que nela, as únicas variáveis são o tempo total T_{total} (eixo y) e a quantidade de blocos a serem validados N (eixo x).

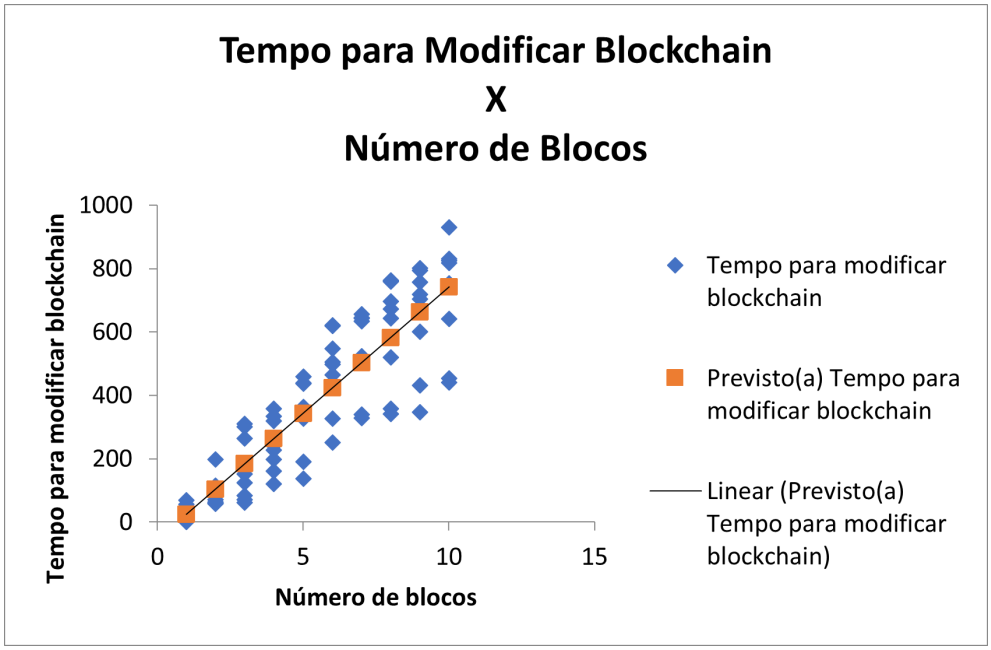

Figura 4.2: Gráfico de tempo necessário para modificar uma blockchain

A segunda etapa de testes consistiu em avaliar o tempo necessário para os mineradores adicionarem alguns arquivos comuns como imagem jpg e vídeo mp4 à blockchain, que poderiam ser utilizados em câmeras de vigilância, dispositivos de comando de voz ou outros dispositivos com telas. Nessa etapa também foi observada a influência do tamanho do bloco imposto pela blockchain e a largura de banda na situação em que um ator solicita esses arquivos da blockchain sem interferência de um man-in-the-middle. Foram realizados 10 testes para cada variação na largura de banda para um mesmo tamanho de bloco e também foram feitos 10 testes de criação para cada tamanho de bloco em cada um dos 6 arquivos usados para os testes. Esses arquivos eram convertidos para base64 a fim de garantir que não fossem corrompidos durante o transporte nas estruturas do protocolo[17] e pudessem ser armazenados como sequências de caracteres. Como foi escolhida essa codificação, houve um aumento de 33% no tamanho dos arquivos, já que nesse formato em vez de 8 bits para representar um caractere, são usados 6 bits para essa representação e, assim, a cada 3 bytes de dados são usados 4 caracteres para representar esses dados em vez do normal de 3 caracteres usados para strings[17]. Depois dessa codificação, esses arquivos eram divididos de modo a não ultrapassar o tamanho máximo do bloco.

Nas Tabelas 4.2 a 4.7, foram identificados o tamanho de cada bloco, número de blocos necessários para guardar todo o arquivo, o tempo médio necessário para criar todos os blocos após 10 testes para cada arquivo da segunda etapa, o tempo de requisição também após 10 testes, o tamanho total de todos os blocos e a largura de banda usada nos

Nº Blocos	Tempo de Requisição (s)	Tempo de Requisição modificada (s)
1	0,005	29.603
2	0,009	88.225
3	0,012	170.49
4	0,006	244.93
5	0,009	337.22
6	0,015	478.88
7	0,015	534.29
8	0,023	593.99
9	0,012	643.98
10	0,079	711.21

Tabela 4.1: Tempo necessário para um man-in-the-middle modificar uma blockchain

testes. Nessas tabelas, é possível notar que mesmo com uma largura baixa de apenas 1 MB/s não é necessário tempos muito elevados, no pior caso 29 MB de blocos foram recebidos em 32 segundos. Além disso, larguras de banda elevadas a partir de 100 MB/s permitem recebimento da blockchain muito rapidamente em tempo inferior a 1 segundo. Desse modo, quando uma aplicação de um ator precisa apenas receber poucos ou pequenos arquivos de uma blockchain ou dados de sensores de microcontroladores, como informação de temperatura, luminosidade, entre outros dados comumente coletados, os fatores mais relevantes são a quantidade de blocos e o tempo necessário para criá-los.

A Figura 4.3 apresenta o tempo previsto para criar os blocos e adicionar na blockchain em relação ao tamanho dos dados quando considerado blocos de 1MB. Nota-se uma tendência linear no aumento de tempo quando o tamanho dos dados é maior. Contudo, observa-se, em conjunto com o tempo de criação nas Tabelas 4.2 a 4.7, que o tempo necessário para armazenar dados na blockchain é elevado e isso dificulta o uso desses dados em tempo real principalmente quando os blocos possuem tamanhos menores como 10KB e 100KB. Em compensação, esses tamanhos por gerarem muito mais blocos com um mesmo arquivo permitem blockchains mais seguras, já que exigem um poder computacional muito superior a blocos de 1MB e 10MB para que um man-in-the-middle consiga modificar os dados.

Comparando-se as duas etapas de testes, é possível observar que o objetivo de prevenir ataques de man-in-the-middle é alcançado com o protocolo baseado em blockchain, já que a necessidade de validar alterações na blockchain prejudica significativamente um invasor e facilita aos atores identificar um ataque por meio do tempo de cada requisição. Entretanto, as aplicações que usam esse protocolo não devem depender de dados em tempo real, é necessário criar os blocos previamente de modo a possuir uma grande quantidade deles a fim de tornar a detecção de invasões confiáveis.

Tamanho Bloco(MB)	Nº blocos	Tempo de Criação(min)	Tempo de Requisição(s)	Tamanho Total(MB)	Largura de Banda(MB/s)
1	1	0.6341	0.082	1	1000
1	1	0.6341	0.085	1	100
1	1	0.6341	0.092	1	50
1	1	0.6341	0.144	1	10
1	1	0.6341	1.127	1	1
10	1	0.8914	0.420	10	1000
0.010	2	1.7298	0.025	0.02055	1000
0.100	1	0.8744	0.029	0.10069	1000

Tabela 4.2: Tempo para criar e obter blocos gerados a partir de uma imagem png de 14,4KB

Tamanho Bloco(MB)	Nº blocos	Tempo de Criação(min)	Tempo de Requisição(s)	Tamanho Total(MB)	Largura de Banda(MB/s)
1	2	1.3034	0.117	2	1000
1	2	1.3034	0.129	2	100
1	2	1.3034	0.131	2	50
1	2	1.3034	0.273	2	10
1	2	1.3034	2.24	2	1
10	1	0.4618	0.729	10	1000
0.010	154	123.12	0.040	1.56	1000
0.100	16	9.6032	0.049	1.57	1000

Tabela 4.3: Tempo para criar e obter blocos gerados a partir de uma imagem jpg de 1.14MB

Tamanho Bloco(MB)	Nº blocos	Tempo de Criação(min)	Tempo de Requisição(s)	Tamanho Total(MB)	Largura de Banda(MB/s)
1	7	5.5054	0.412	7	1000
1	7	5.5054	0.424	7	100
1	7	5.5054	0.446	7	50
1	7	5.5054	0.964	7	10
1	7	5.5054	8.19	7	1
10	1	0.7829	0.577	10	1000
0.010	689	524,83	0.193	6.97	1000
0.100	69	57.038	0.122	6.76	1000

Tabela 4.4: Tempo para criar e obter blocos gerados a partir de uma imagem png de 5.05MB

Tamanho Bloco(MB)	Nº blocos	Tempo de Criação(min)	Tempo de Requisição(s)	Tamanho Total(MB)	Largura de Banda(MB/s)
1	27	22.5318	1.546	27.01	1000
1	27	22.5318	1.452	27.01	100
1	27	22.5318	1.606	27.01	50
1	27	22.5318	3.52	27.01	10
1	27	22.5318	31.07	27.01	1
10	3	3.0562	1.412	30	1000
0.010	2744	2149.90	0.941	27.78	1000
0.100	275	232.109	0.419	26.95	1000

Tabela 4.5: Tempo para criar e obter blocos gerados a partir de um vídeo mp4 de 20MB

Tamanho Bloco(MB)	Nº blocos	Tempo de Criação(min)	Tempo de Requisição(s)	Tamanho Total(MB)	Largura de Banda(MB/s)
1	7	5.4948	0.123	7	1000
1	7	5.4948	0.185	7	100
1	7	5.4948	0.241	7	50
1	7	5.4948	0.833	7	10
1	7	5.4948	7.78	7	1
10	1	0.8023	0.160	10	1000
0.010	689	572.19	0.213	6.97	1000
0.100	69	53.834	0.117	6.76	1000

Tabela 4.6: Tempo para criar e obter blocos gerados a partir de um áudio mp3 de 5MB

Tamanho Bloco(MB)	Nº blocos	Tempo de Criação(min)	Tempo de Requisição(s)	Tamanho Total(MB)	Largura de Banda(MB/s)
1	29	23.6044	0.370	29	1000
1	29	23.6044	0.693	29	100
1	29	23.6044	0.963	29	50
1	29	23.6044	3.48	29	10
1	29	23.6044	32.11	29	1
10	3	3.5812	0.435	30	1000
0.010	2918	2354.29	0.929	29.54	1000
0.100	292	234.793	0.583	28.62	1000

Tabela 4.7: Tempo para criar e obter blocos gerados a partir de um áudio flac de 21.37MB

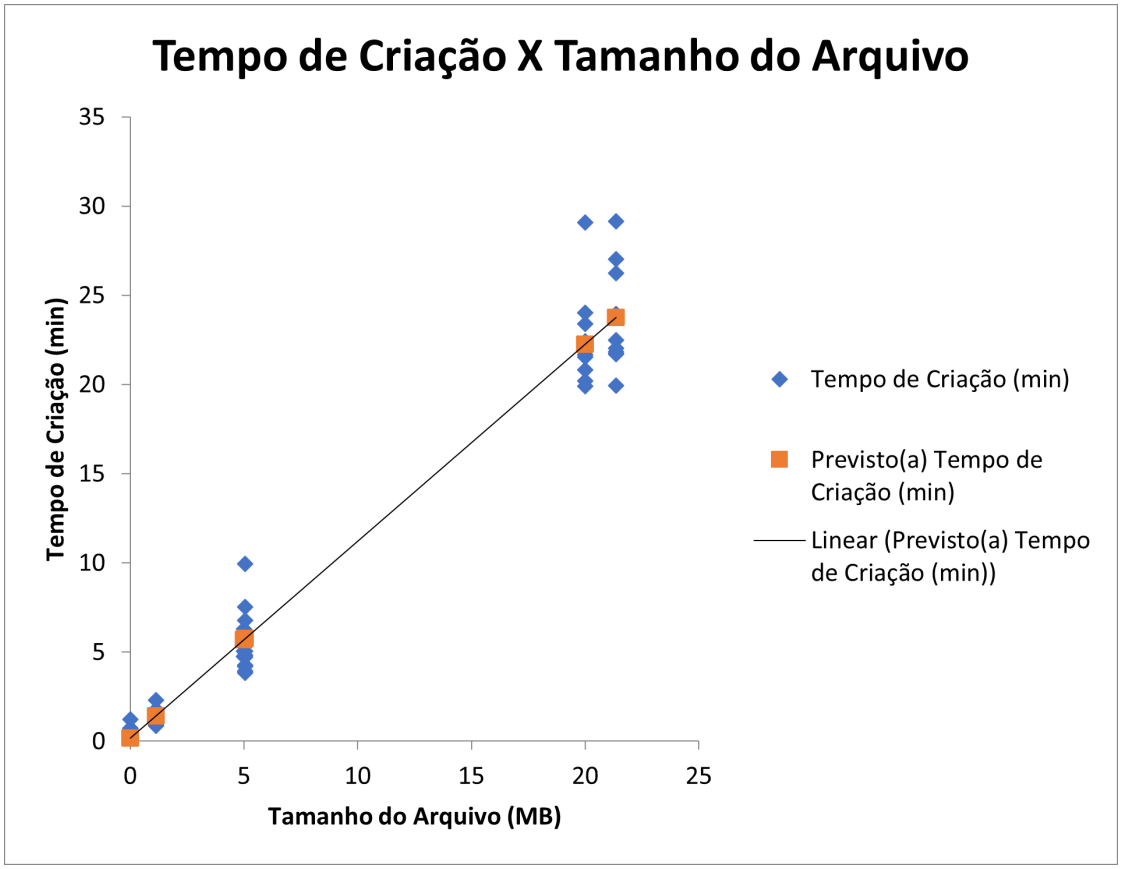

Figura 4.3: Gráfico de tempos em relação ao tamanho do arquivo

Capítulo 5

Conclusões

O objetivo do trabalho era obter um mecanismo para dispositivos IoT se comunicarem de forma segura, visto a falta de módulos de criptografia em microcontroladores comumente utilizados na indústria. Para isso, o interesse do protocolo era impedir que um MITM pudesse interferir no funcionamento desses dispositivos alterando os dados transmitidos. Os resultados obtidos mostram que o uso de uma blockchain torna a tarefa de alterar os dados muito difícil ao exigir um tempo significativo durante a transmissão de dados para isso ser feito. Desse modo, é possível com o método aumentar a confiança dos dados que aplicativos e sistemas receberão de dispositivos IoT. Entretanto, o protocolo apresenta algumas limitações, como a exigência de uma placa de processamento gráfico e dispositivo com grande capacidade de armazenamento no ambiente interno onde os microcontroladores fazem a coleta de dados. Além disso, o armazenamento na blockchain é um processo lento que pode exigir minutos a depender das informações guardadas. Isso prejudica o uso em tempo real e também a criação de um ambiente composto somente por dispositivos de baixa capacidade de processamento. Diante dessas limitações, trabalhos futuros podem verificar a possibilidade de substituir o algoritmo de consenso utilizado (prova de trabalho) por outro, como o prova de participação, o qual não exige grande poder computacional. Também, pode-se analisar os benefícios de associar este protocolo com outros, já que não há restrição nos dados armazenados na blockchain.

Referências

[1] Nakamoto, Satoshi e A Bitcoin: *A peer-to-peer electronic cash system*. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf, 4, 2008. 2

[2] Keim, Robert: *What is a microcontroller? the defining characteristics and architecture of a common component*. `https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component`, acesso em 2019-07-01. 4

[3] Hyncica, Ondrej, Pavel Kucera, Petr Honzik e Petr Fiedler: *Performance evaluation of symmetric cryptography in embedded systems*. Em *Proceedings of the 6th IEEE international conference on intelligent data acquisition and advanced computing systems*, volume 1, páginas 277–282. IEEE, 2011. 4

[4] Sultan, Karim, Umar Ruhi e Rubina Lakhani: *Conceptualizing blockchains: characteristics & applications*. arXiv preprint arXiv:1806.03693, 2018. 4

[5] IBM: *O que é a tecnologia blockchain?* `https://www.ibm.com/br-pt/topics/what-is-blockchain`, acesso em 2021-04-26. 5

[6] Wagner, Lane: *How sha-256 works step-by-step*. `https://qvault.io/cryptography/how-sha-2-works-step-by-step-sha-256/`, acesso em 2021-04-09. 6

[7] Rachmawati, D, J T Tarigan e A B C Ginting: *A comparative study of message digest 5(MD5) and SHA256 algorithm*. Journal of Physics: Conference Series, 978:012116, mar 2018. `https://doi.org/10.1088/1742-6596/978/1/012116`. 6

[8] IV, Robert Greenfield: *Proof of stake*. `https://robertgreenfieldiv.medium.com/explaining-proof-of-stake-f1eae6feb26f`, acesso em 2021-04-16. 6

[9] Saad, Muhammad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, Daehun Nyang e David Mohaisen: *Exploring the attack surface of blockchain: A comprehensive survey*. IEEE Communications Surveys Tutorials, PP:1–1, março 2020. 6, 7

[10] Kraken: *Proof of work vs proof of stake*. `https://www.kraken.com/pt-br/learn/proof-of-work-vs-proof-of-stake`, acesso em 2021-04-16. 6, 7

[11] Binance: *What is a blockchain consensus algorithm.* https://academy.binance.com/pt/articles/what-is-a-blockchain-consensus-algorithm, acesso em 2021-04-09. 6, 7

[12] Ogino, Ozora, Alex Ismodes, Paul Wackerow, Ryan Cordell, Alwin Stockinger e Sam Richards: *Ethereum 2.0 proof of stake.* https://ethereum.org/pt-br/developers/docs/consensus-mechanisms/pos/, acesso em 2021-04-26. 7

[13] Bitcoin, Guia do: *Proof of stake guia do bitcoin.* https://guiadobitcoin.com.br/pos-protocolo-proof-of-stake/, acesso em 2021-04-16. 7

[14] Docker: *Documentação do docker.* https://docs.docker.com/get-started/overview/, acesso em 2021-04-26. 8

[15] Cekerevac, Zoran, Zdenek Dvorak, L. Prigoda e Petar Čekerevac: *Internet of things and the man-in-the-middle attacks – security and economic risks.* MEST Journal, 5:15–5, julho 2017. 10

[16] Adnane, Asma, Farhan Ahmad, Virginia Franqueira, Fatih Kurugollu e Lu liu: *Man-in-the-middle attacks in vehicular ad-hoc networks: Evaluating the impact of attackers' strategies.* Sensors, 18, novembro 2018. 10

[17] Contributors, MDN: *Base64.* https://developer.mozilla.org/en-US/docs/Glossary/Base64, acesso em 2021-04-29. 26