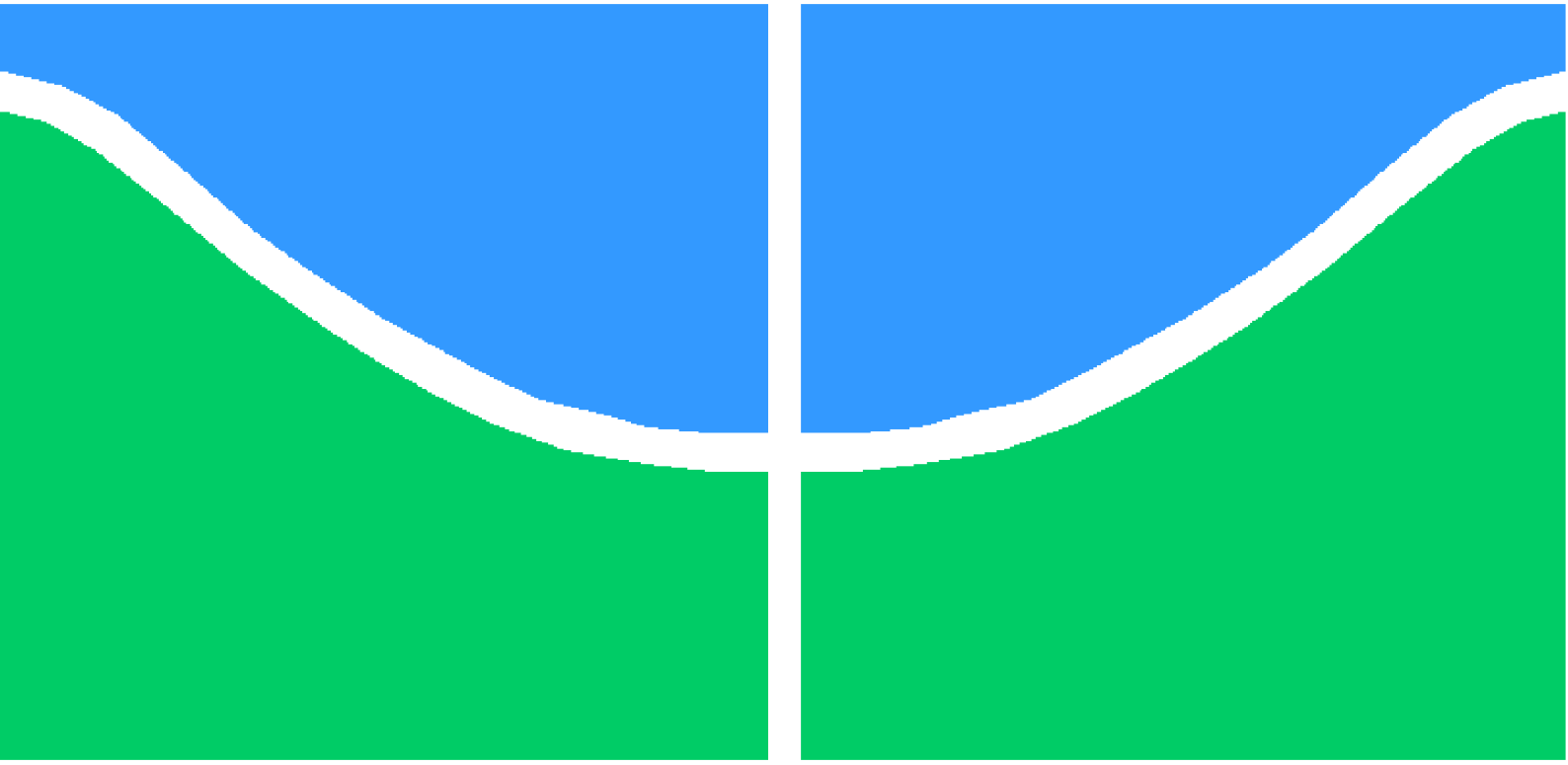


Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Uma aplicação Blockchain para o registro de infrações de trânsito

Autor: João Paulo Nunes Soares
Orientador: Prof. Dr. Tiago Alves da Fonseca

Brasília, DF
2020

João Paulo Nunes Soares

Uma aplicação Blockchain para o registro de infrações de trânsito

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Tiago Alves da Fonseca

Brasília, DF

2020

João Paulo Nunes Soares
Uma aplicação Blockchain para o registro de infrações de trânsito/ João Paulo Nunes Soares. – Brasília, DF, 2020-
74 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Tiago Alves da Fonseca

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2020.

1. Blockchain. 2. Smart Contracts. I. Prof. Dr. Tiago Alves da Fonseca. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Uma aplicação Blockchain para o registro de infrações de trânsito

CDU

João Paulo Nunes Soares

Uma aplicação Blockchain para o registro de infrações de trânsito

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Prof. Dr. Tiago Alves da Fonseca
Orientador

Dr. John Lenon Cardoso Gardenghi
Convidado 1

Me. José Roberto Menezes Monteiro
Convidado 2

Brasília, DF
2020

Agradecimentos

Gostaria de agradecer a todos que diretamente ou indiretamente fizeram parte da minha formação, não só dentro da Universidade de Brasília assim como de todos os outros lugares que fazem parte da minha história.

Agradeço também de forma especial ao Prof. Dr. Tiago Alves da Fonseca não só por todo o aprendizado e auxílio durante o desenvolvimento deste trabalho mas também por todo conhecimento que compartilhou de forma exemplar e inspiradora durante as disciplinas que tive a oportunidade de ser discente.

E agradeço a minha família por sempre me apoiar e estar presente, por serem o exemplo que levarei durante toda a minha vida. Sem eles não poderia estar aqui hoje escrevendo este trabalho e por eles vejo um futuro a ser vivido.

Resumo

O termo Blockchain se popularizou em função das negociações com criptomoedas, de forma mais notória a moeda digital Bitcoin que surgiu em meados de 2009, porém sua tecnologia possui um potencial de aplicação em diversos contextos. Este potencial de uso da tecnologia fez com que diversas plataformas e comunidades de apoio ao desenvolvimento destas aplicações surgissem, possibilitando que diversas aplicações fossem criadas e a tecnologia difundida cada vez mais em diversas áreas. Os registros feitos dentro de um blockchain são imutáveis, uma vez que estes registros são posicionados em blocos e vinculados a registros anteriores após o consenso de outros nós presentes na rede, o que torna não necessária a presença de uma terceira parte para autenticar a veracidade das informações apresentadas. O presente trabalho propõe a implementação de uma aplicação descentralizada (Dapp) para o registro de infrações de trânsito de acordo com as normas presentes no Código de Trânsito Brasileiro, demonstrando a viabilidade do uso desta nova tecnologia em um contexto presente no cotidiano das pessoas.

Palavras-chaves: Blockchain. Contratos Inteligentes. Ethereum. Infrações de Trânsito.

Abstract

Blockchain became popular due to the widespread use of cryptocoins. The Bitcoin cryptocoin was introduced in mid-2009 and offered a real test field for blockchain because Bitcoin relies on it to digitally provide the same features of a regular currency. However, it should be noted that the blockchain technology has a potential of application in different context. This potential use of blockchain technology has stimulated various platforms and communities to support the development of these applications, allowing various applications to be created and to spread the technology in many areas. The records made in a blockchain are immutable, and these records are placed in blocks linked to previous records after the consensus of other nodes present in the network, which does not require the presence of a third party to authenticate the authenticity of the presented information. The present work proposes the implementation of a decentralized application (Dapp) for the registration of traffic infractions according to the norms present in the Brazilian Traffic Code, demonstrating the feasibility of the use of this new technology in a context of the daily life.

Key-words: Blockchain. Smart Contracts. Ethereum. Traffic Infractions.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Não resta dúvida de que atualmente o trânsito seria inviável sem a presença de normas, de regras que regulam a movimentação e a ocupação do espaço viário. Toda a supervisão do trânsito (fiscalização, controle) está baseada num conjunto de leis e dispositivos, que constam nos códigos e regulamentos de cada país. Em toda a experiência mundial de trânsito, as normas juntamente com a fiscalização do seu cumprimento têm exercido um papel fundamental, na medida em que é muito notável a relação direta entre fiscalização-punição e comportamento adequado (não infração às normas) com a diminuição de acidentes, e inversamente entre a impunidade e o comportamento inadequado (infração às normas) com aumento de acidentes. Um sistema normativo é condição indispensável para a fluidez e segurança no trânsito, sendo esta, sempre relacionada (e medida) aos índices de acidentes (HOFFMANN MARIA HELENA; CARBONELLI, 1996).

Em 2018 foram registradas pelas autoridades de trânsito somente no Distrito Federal um total 2.740.685 infrações de trânsito, o que corresponde a uma média diária de aproximadamente 7500 infrações. Este número se torna ainda maior quando considerado os registros realizados por órgãos de outras unidades federativas, e também pelos órgãos responsáveis por fiscalizações em âmbito federal como a Polícia Rodoviária Federal.

A forma e o sistema em que o registro da infração de trânsito é feito depende de fatores como o órgão autuador e a localização geográfica da mesma. Esse forma de registro não padronizada, centralizada pelas autoridades e condicionado a diversos fatores fazem com que os serviços associados aos órgãos de trânsito, como aplicação de multas ou realização de possíveis recursos, se tornem ineficientes e lentos, o que eleva o custo de manutenção para o Estado e aumenta a insatisfação de todos os cidadãos que precisam utilizar algum destes serviços.

De forma a integrar esta capilaridade de sistemas presentes nas Unidades Federativas, o sistema RENAINF (Registro Nacional de Infrações de Trânsito) foi criado e é atualmente coordenado pelo DENATRAN (Departamento Nacional de Trânsito). Porém este sistema somente é utilizado para registrar as infrações de trânsito cometidas em unidade federada diferente daquela de onde o veículo estiver registrado e licenciado, bem como para o registro das infrações impostas pelas autoridades de trânsito federadas independente da vinculação de registro do veículo. Este sistema possibilita que o órgão autuador obtenha os dados necessários para registrar a informação da infração cometida e vincular estes débitos no Departamento Estadual de Trânsito (DETRAN) de registro do veículo, não sendo assim um sistema realmente unificado para o registro de infrações de trânsito no território brasileiro (FAZENDA E PLANEJAMENTO DE SãO PAULO,

2018).

A construção de um sistema nacional único e centralizado por uma autoridade como o DENATRAN poderia mitigar problemas decorrentes dessa capilaridade de sistemas, porém esta centralização acaba tendo seus pontos negativos como um maior custo de manutenção e maior dificuldade para garantir a integridade do sistema. Em meados de 2009 uma nova tecnologia denominada Blockchain foi proposta para o uso em uma aplicação monetária que não necessitava de uma unidade central de confiança para seu funcionamento, e logo observou-se que seu conceito poderia ser empregado em contextos diferentes do que foi idealizado por possibilitar algumas vantagens como a eliminação de intermediários e o aumento da segurança com custo baixo (PANDA, 2018).

1.1 Justificativa

Blockchain é uma tecnologia que possui um grande potencial de utilização em diversos setores da sociedade e esta revolução já começou em algumas áreas, afetando enormemente o mercado de serviços financeiros. É difícil nomear um banco global ou uma entidade financeira que não explore o blockchain. Além do mercado financeiro, iniciativas já estão sendo tomadas em áreas como mídia e entretenimento, comercialização de energia, mercados de previsão, redes de varejo, sistemas de fidelidade, seguro, logística e cadeias de suprimento, registros médicos e também aplicações governamentais e militares (PANDA, 2018).

Considerando o potencial para aplicação desta tecnologia disruptiva, principalmente para aplicação em serviços considerados de ordem pública como os oferecidos e mantidos pelo Governo, e a existência de problemáticas como a elencada anteriormente em relação ao registro de infrações de trânsito este trabalho tem como proposta demonstrar a viabilidade da construção e utilização de uma aplicação descentralizada em um contexto hoje centralizado.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo principal da pesquisa é desenvolver uma aplicação descentralizada para o registro de infrações de trânsito, assim como possibilitar por meio da ferramenta a realização de outros serviços previstos pelo CTB.

1.2.2 Objetivos Específicos

1. Desenvolver uma aplicação descentralizada acessível a todos que desejarem;

2. Estudar sobre a nova tecnologia *blockchain* e como ela pode ser usada em diversas áreas;

3. Demonstrar a viabilidade do uso de aplicações descentralizadas em um contexto atualmente centralizado;

1.3 Estrutura do Documento

Este documento está composto pelos seguintes capítulos:

1. **Introdução:** A introdução contém um breve contexto e motivações para a realização deste trabalho;

2. **Fundamentação Teórica:** A fundamentação teórica contém conceitos teóricos considerados necessários para uma melhor compreensão do trabalho que será desenvolvido;

3. **Proposta de Trabalho:** Contém uma explanação da proposta de trabalho à ser desenvolvido;

4. **Metodologia:** Contém descrição da forma que o projeto será desenvolvido, assim como metodologias, políticas e ferramentas que serão utilizadas.

5. **Resultados e Discussões:** Contém uma explanação da solução desenvolvida, contendo informações relacionadas as ferramentas utilizadas, arquitetura e considerações.

6. **Conclusão:** Contém as considerações finais a cerca do trabalho desenvolvido

2 Fundamentação Teorica

2.1 Conceitos de segurança

O Computer Security Handbook(NIST, 1995), define segurança computacional como:

> "A proteção oferecida a um sistema de informações automatizado para atingir os objetivos aplicáveis de preservar a integridade, a disponibilidade e a confidencialidade dos recursos do sistema de informações (incluindo *hardware*, *software*, *firmware*, informações / dados e telecomunicações)."

STALLINGS escreve em seu livro que esta definição introduz os três objetivos chaves da segurança computacional, que estão descritos a seguir:

- **Confidencialidade:** assegura que uma informação privada ou confidencial não estará disponível para indivíduos não autorizados, e que o indivíduo tem controle sob suas informações;
- **Integridade:** assegura que as informações e as aplicações são modificadas somente por indivíduos autorizados, e que os sistemas sejam executados de forma que não haja manipulações por terceiros;
- **Disponibilidade:** Assegura que os sistemas estão prontamente sendo executados e que os serviços não são negados para usuários autorizados.

2.1.1 Criptografia

A criptografia é o componente mais importante do *blockchain*. A criptografia é certamente um campo de pesquisa em si mesmo, e é baseada em técnicas matemáticas. Houve muitas falhas relatadas em carteiras, seção 2.5, e trocas de informações devido ao *design* mais fraco ou implementações criptográficas frágeis (PANDA, 2018).

Qualquer informação na forma de uma mensagem de texto, dados numéricos ou um programa de computador pode ser chamada de texto claro. A ideia é cifrar o texto claro usando um algoritmo de cifração/decifração e uma chave que produza o texto cifrado, este último se tornando uma versão protegida do texto claro em relação a confidencialidade. O texto cifrado pode então ser transmitido ao destinatário pretendido, que o decifra usando o algoritmo de decifração e a chave para obter o texto simples.(PANDA, 2018)

Em geral, existem dois tipos de criptografia: criptografia de chave simétrica e criptografia de chave assimétrica. A seguir será explicado cada um desses tipos.

2.1.1.1 Modelo de criptografia simétrica

Um sistema de criptografia simétrica é um sistema criptográfico no qual a cifração e a decifração são executadas usando a mesma chave. Também é conhecido como criptografia convencional. A criptografia simétrica transforma o texto claro em texto cifrado usando uma chave secreta e um algoritmo de cifração. Usando a mesma chave e um algoritmo de decifração, o texto original é recuperado do texto cifrado (STALLINGS, 2011).

Assume-se que é impraticável decifrar uma mensagem com base no texto cifrado e no conhecimento do algoritmo de cifração/ decifração. Em outras palavras, não é necessário manter o algoritmo secreto; precisa-se manter apenas o segredo da chave. Esse recurso de criptografia simétrica é o que o torna viável para uso generalizado. Com o uso de criptografia simétrica, o principal problema de segurança é manter o sigilo da chave (STALLINGS, 2011).

A Figura 1 representa como é o funcionamento da criptografia simétrica. Na figura, existe a diferenciação de dois canais de comunicação, onde o canal considerado inseguro é utilizado para transmitir o texto que foi cifrado sem prejuízos a sua segurança; e um canal considerado seguro para transmitir a chave utilizada para cifrar o texto e que também será utilizada para decifrar o mesmo e recuperar o texto em claro.

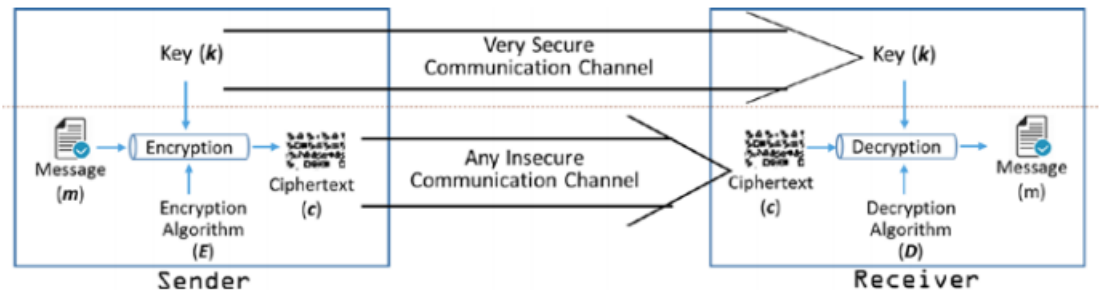
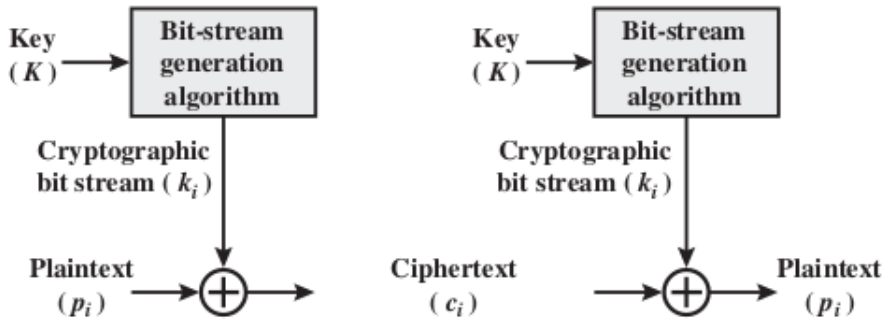


Figura 1 – Criptografia Simétrica - Imagem retirada de (PANDA, 2018).

A criptografia de chave simétrica existe em duas variantes: *stream ciphers* e *block ciphers* (PANDA, 2018).

Um *stream ciphers* ou **cifra de fluxo** é a variante que cifra um fluxo de dados digital, um bit ou um byte de cada vez. Se o fluxo de chaves criptográficas for aleatório, essa cifra é inquebrável por qualquer outro meio que não seja a aquisição do fluxo de chaves. No entanto, o fluxo de chaves deve ser fornecido aos dois usuários antecipadamente por meio de um canal independente e seguro. Isso introduz problemas logísticos insuperáveis se o tráfego de dados pretendido for muito grande (STALLINGS, 2011).

Consequentemente, por razões práticas, o gerador de fluxo de bits deve ser implementado como um procedimento algorítmico, de modo que o fluxo de bits criptográfico possa ser produzido por ambos os usuários. Nesta abordagem (Figura 2), o gerador de fluxo de bits é um algoritmo controlado por chave e deve produzir um fluxo de bits criptograficamente fortes. Agora, os dois usuários precisam compartilhar apenas a chave geradora, e cada um pode produzir o fluxo de chaves (STALLINGS, 2011).



(a) Stream cipher using algorithmic bit-stream generator

Figura 2 – Cifrador de fluxo usando algoritmo gerador de fluxo de bits - Imagem retirada de (STALLINGS, 2011).

Um *block cipher* ou **cifra de bloco** é aquela em que um bloco de texto simples é tratado como um todo e usado para produzir um bloco de texto cifrado de comprimento igual. Normalmente, um tamanho de bloco de 64 ou 128 bits é usado. Caso o texto a ser cifrado seja maior que o tamanho do bloco utilizado, o texto é dividido em blocos do tamanho suportado e cada um dos blocos é cifrado utilizando a chave. O último bloco pode sofrer o processo de padding caso o seu tamanho seja menor que o tamanho de bloco utilizado para cifração.

Como ocorre com uma cifra de fluxo, os dois usuários compartilham uma chave de criptografia simétrica (Figura 3).

2.1.1.2 Modelo de criptografia assimétrico

Criptografia de chave assimétrica, também conhecida como criptografia de chave pública, é um conceito introduzido por Diffie e Hellman. Com essa técnica, eles resolveram o problema da distribuição de chaves para um sistema de criptografia simétrica, onde distribuição desta chave única precisava ser feita por meio de um canal extramente seguro de transmissão que era por muitas vezes sendo entregue ou repassada pessoalmente, introduzindo assim também a possibilidade da criação de assinaturas digitais (PANDA, 2018).

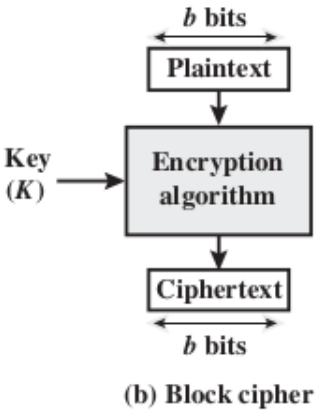


Figura 3 – Cifrador de Bloco- Imagem retirada de (STALLINGS, 2011).

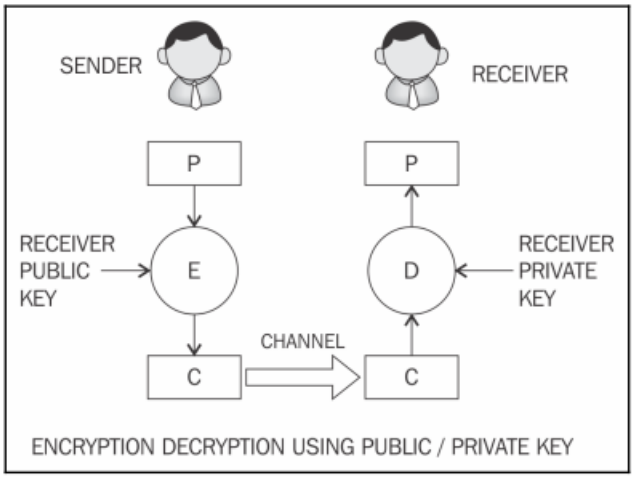
Antes de prosseguir, deve-se falar sobre um grande equívoco comum em relação a criptografia de chave pública. Esse equívoco é que a criptografia de chave pública é mais segura do que a criptografia simétrica. De fato, a segurança de qualquer esquema de criptografia depende do tamanho da chave e do trabalho computacional envolvido em quebrar uma cifra. Não há nada em princípio sobre criptografia simétrica ou de chave pública assimétrica que a torne superior a outra do ponto de vista da resistência à criptoanálise (STALLINGS, 2011).

A criptografia assimétrica é um sistema criptográfico na qual a cifração e a decifração são realizadas utilizando diferentes chaves - uma chave pública e uma chave privada. A criptografia assimétrica transforma o texto claro em texto cifrado usando uma das duas chaves e um algoritmo de criptografia. Usando a chave emparelhada e um algoritmo de decifração, o texto original é recuperado a partir do texto cifrado. A criptografia assimétrica pode ser usada para confidencialidade, autenticação, ou ambos (STALLINGS, 2011).

É importante ressaltar que a diferença na criptografia assimétrica não é somente relacionado ao uso de diferentes chaves, porém também na utilização de algoritmos de cifração e decifração diferentes dos utilizados na criptografia simétrica.

A Figura 4 explica como um remetente cifra o texto claro usando a chave pública de um destinatário, que gera o texto cifrado, que será transmitido pela rede para o receptor. Uma vez que chegue ao receptor, o texto pode ser decifrado usando a chave privada do receptor. Desta forma, a chave privada permanece no lado do receptor e não há necessidade de compartilhar chaves, a fim de realizar a cifração e a decifração, que é o caso da criptografia simétrica (BASHI, 2017).

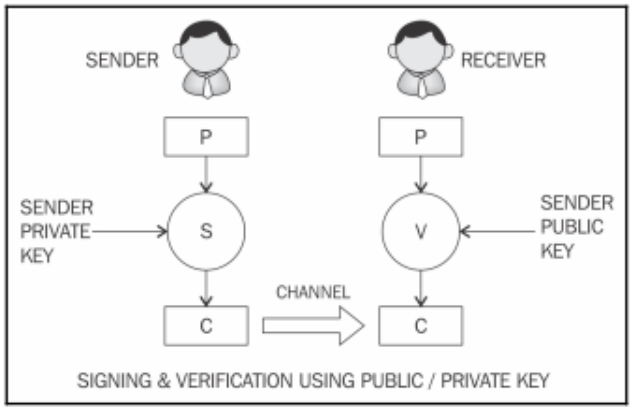
A Figura 5 mostra como a criptografia de chave pública pode ser usada para verificar a autenticidade da mensagem recebida pelo receptor. Nesse modelo, o remetente assina os dados usando sua chave privada e transmite a mensagem para o receptor. Quando a mensagem é recebida no lado do receptor, pode ter sua autenticidade verificada através



Encryption decryption using public/private key

Figura 4 – Criptografia e descriptografia usando chave público/privada - Imagem retirada de (BASHI, 2017).

do processamento com a chave pública do remetente. Observe que não há busca de confidencialidade/sigilo neste modelo (BASHI, 2017). Este modelo é usado apenas para fins de autenticação e validação de mensagem.



Model of a public key cryptography signature scheme

Figura 5 – Modelo de esquema assinatura por criptografia de chave pública - Imagem retirada de (BASHI, 2017).

2.1.2 Funções Hash

Funções hash são funções criptográficas primitivas matemáticas e são parte integrante da estrutura fundamental do blockchain. Elas são amplamente utilizadas em muitos protocolos criptográficos, aplicativos de segurança da informação, como assinaturas digitais e códigos de autenticação de mensagens (MACs) (PANDA, 2018).

Uma função hash H aceita um bloco de comprimento variável de dados M como entrada e produz um valor de hash de tamanho fixo h = H (M). Uma boa função de hash tem a propriedade de que os resultados da aplicação da função a um grande conjunto de entradas produzirá saídas uniformemente distribuídas e aparentemente aleatórias. Em termos gerais, o objetivo principal de uma função hash é a integridade dos dados. Uma mudança em qualquer bit ou bits em M resulta, com alta probabilidade, em uma mudança no hash calculado (STALLINGS, 2011).

O tipo de função hash necessária para aplicativos de segurança é chamada de **função hash criptográfica**. Uma função hash criptográfica é um algoritmo para o qual é computacionalmente inviável (porque nenhum ataque é significativamente mais eficiente que a força bruta) para encontrar (a) um objeto de dados que mapeia para um resultado de hash pré-especificado (a propriedade unidirecional) ou (b) dois objetos de dados que mapeiam para o mesmo resultado de hash (a propriedade de ser livre de colisão). Devido a essas características, as funções hash costumam ser usadas para determinar se os dados foram alterados ou não (STALLINGS, 2011).

A Figura 6 mostra o funcionamento geral de uma função hash criptográfica. Normalmente, a entrada é preenchida com um inteiro múltiplo de algum comprimento fixo (por exemplo, 1024 bits) e o preenchimento inclui o valor do comprimento da mensagem original em bits. O campo de comprimento é uma medida de segurança para aumentar a dificuldade de um invasor para produzir uma mensagem alternativa com o mesmo valor de hash (STALLINGS, 2011).

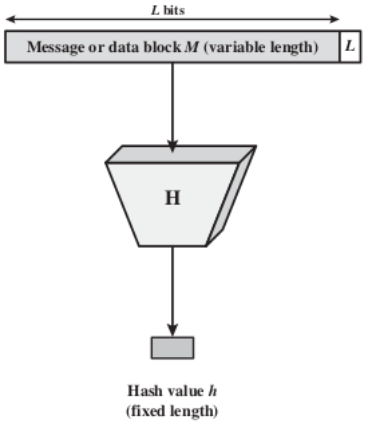


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

Figura 6 – Diagrama de função de hash criptográfica; h = H(M) - Imagem retirada de (STALLINGS, 2011).

Uma das mais antigas funções de hash, ou função de compressão, é a função hash MD4, que pertence à família message digest (MD). Entre os membros da família MD temos MD5 e MD6, e muitas outras variantes do MD4, como o RIPEMD. A família de algoritmos MD produz um resumo de mensagens de 128 bits consumindo blocos de 512

bits. Eles foram amplamente utilizados como checksums para verificar a integridade dos dados (PANDA, 2018).

Outra família de funções hash amplamente utilizada é a **Secure Hash Algorithm (SHA)**. Existem basicamente quatro algoritmos nessa família: SHA-0, SHA-1, SHA-2 e SHA-3 (PANDA, 2018).

É importante ressaltar que alguns destes algoritmos como o MD, SHA-0 e o SHA-1 hoje são considerados não seguros, pois é possível que a colisão seja provocada com menos tentativas que o espero inicialmente pelo algoritmo. O trabalho de WANG YIQUN LISA YIN exemplifica este caso na quebra do algoritmo SHA1.

2.1.2.1 Ataques em funções de hash

O uso de funções Hash é bem conhecido por pessoas que estão trabalhando na área de criptografia, ou seja, ramo de rede e segurança, pois o Hash funciona adequadamente fornecendo serviços de segurança necessários por um longo período de tempo até hoje e não é possível quebrá-lo facilmente, mas ainda existem grandes problemas que direta ou indiretamente ajudam o(s) cracker(s) a quebrar a segurança das funções hash ou funções de compressão até certo ponto.(SHARMA; S.K.MITTAL, 2018)

SHARMA; S.K.MITTAL descreve em seu artigo alguns testes tipos de ataques possíveis relacionados a funções de hash, a seguir alguns destes são elencados assim como uma breve explanação sobre os mesmos feita pelo autor.

- **Brute Force Attack:** O ataque de força bruta é o único ataque que realmente funciona em todas as funções hash criptográficas, é a busca exaustiva (por mensagem), ou seja, todas as formas possíveis de percorrer os algoritmos com a ajuda de saída (resumo). Como nos sistemas de criptografia-descriptografia criptográfica, esse ataque encontra a chave secreta, da mesma forma que aqui ele encontra a mensagem real a partir da qual o resumo foi gerado pelo respectivo algoritmo de comprimento desejado.

- **Birthday Attack (Birthday Pardox):**O Ataque de aniversário é mais especificamente usado para localizar colisões em funções de hash criptográficas. O trabalho desse ataque é descrito diretamente a seguir: Seja 'h' uma função hash de n bits (tamanho do resumo) com mensagem de entrada legítima (m1) e mensagem de falsificação (m2). Portanto, a saída será (m'1), (m'2) resultante de pequenas modificações de (m1) e (m2) com H (m'1) = H (m'2). Portanto: (1) Gere 'N' = 2n / 2 modificações menores (m'1) de (m1). (2) Compute o hash de cada uma dessas mensagens e armazene o(s) resultado(s) de hash. (3) Gere modificações menores (m'2) de (m2), calculando H (m'2) para cada um e verificando as correspondências com qualquer H (m'1) da etapa anterior, continue até que a correspondência seja

encontrada. Uma correspondência pode ser esperada após cerca de 'N' candidatos de (m'2);

- **Boomerang Attack:** O ataque bumerangue foi introduzido para reduzir a complexidade da Criptoanálise Diferencial. Neste ataque para análise, em vez de usar apenas uma característica diferencial longa com baixa probabilidade, estamos usando duas características diferenciais curtas com alta probabilidade. Para este efeito, a cifra de bloco 'E' é tratada como uma cascata de duas subcifras 'E0' e 'E1' (ou seja, $E = E1 \circ E0$). O ataque de bumerangue pode ser aplicado à função interna e versões reduzidas de muitas funções hash.

2.1.2.2 Ponteiros Hash

Um **ponteiro de hash (hash pointer)** é um hash criptográfico apontando para um bloco de dados, onde o ponteiro de hash é o hash do próprio bloco de dados (Figura 7). Ao contrário das listas vinculadas que apontam para o próximo bloco para que você possa chegar até ele, os ponteiros de hash apontam para o bloco de dados anterior e fornecem uma maneira de verificar se os dados não foram adulterados (PANDA, 2018).

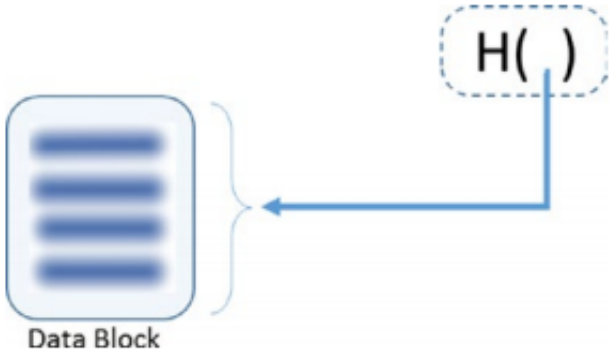


Figura 7 – Ponteiro Hash para um bloco de dados - Imagem retirada de (PANDA, 2018).

O objetivo do ponteiro de hash é construir uma cadeia de blocos que pode ser considerada como uma fonte única de verdade. Seu funcionamento se baseia no fato que o hash do bloco anterior é armazenado no cabeçalho do bloco atual e o hash do bloco atual com o cabeçalho do bloco será armazenado no cabeçalho do próximo bloco. Isso cria uma cadeia de blocos como podemos ver na Figura 8 (PANDA, 2018).

2.1.2.3 Merkle Trees

Uma Árvore Merkle, também conhecida como uma árvore hash binária, é uma estrutura de dados usada para resumir e verificar eficientemente a integridade de grandes conjuntos de dados. Árvores Merkle são árvores binárias contendo hashes criptográficos. O termo árvore é usado na ciência da computação para descrever uma estrutura de dados

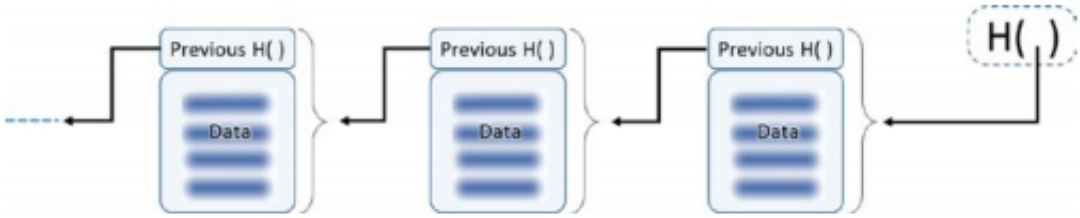


Figura 8 – Blocos encadeados ligados por ponteiros hash - Imagem retirada de (PANDA, 2018).

de ramificação, mas essas árvores geralmente são exibidas de cabeça para baixo com a raiz na parte superior e as folhas na parte inferior de um diagrama.(ANTONOPOULOS, 2017)

Uma típica Árvore Merkle pode ser representada como na Figura 9.

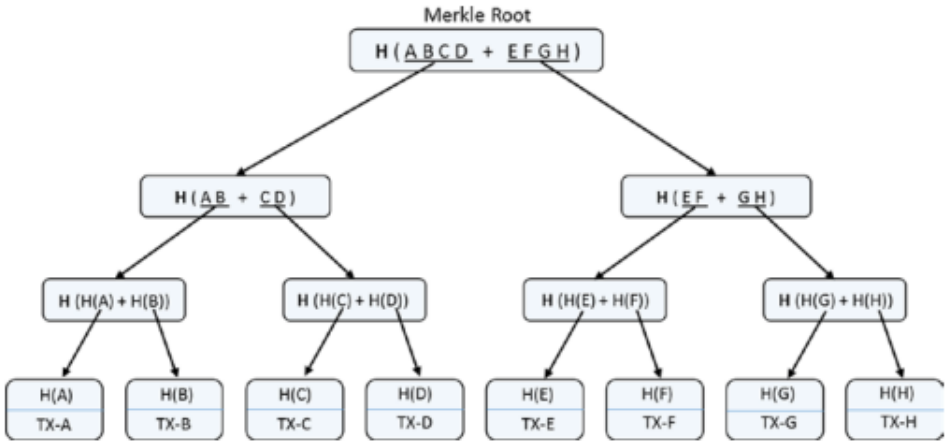


Figura 9 – Representação de uma Árvore de Merkle - Imagem retirada de (PANDA, 2018)

Semelhante à estrutura de dados do ponteiro hash, a árvore Merkle também é resistente à adulteração. A adulteração em qualquer nível da árvore não corresponderia ao hash armazenado em um nível acima na hierarquia e também até o nó raiz. É realmente difícil para um adversário alterar todos os hashes da árvore inteira. Também garante a integridade da ordem das transações. Se você alterar apenas a ordem das transações, também os hashes na árvore até a raiz Merkle serão alterados.(PANDA, 2018)

2.2 Sistemas Distribuídos

Diversas definições de sistemas distribuídos são apresentadas na literatura, onde estas variam de acordo com o contexto em que esta é apresentada. Para o propósito deste trabalho, pode-se dizer que: “Um sistema distribuído é uma coleção de elementos computacionais autônomos que para os usuários parecem ser um único sistema coerente.

Um elemento computacional, normalmente chamado de nó, pode ser tanto um dispositivo de hardware ou um processo de software" (TANENBAUM ANDREW S.; STEEN, n.d.).

Um nó pode ser definido como um "jogador individual", onde em um sistema distribuído todos os nós tem a capacidade de enviar e receber mensagem entre eles. Estes nós podem ser honestos, faltantes ou maliciosos e podem ter sua própria memória e processador. Um nó que apresenta um comportamento arbitrário também é chamado de Nó Bizantino (BASHI, 2017).

O maior desafio em sistemas distribuídos é a coordenação entre esses nós e a tolerância a falha. Pois mesmo que um determinado nó apresente uma falha de conexão durante um período de tempo ou caso um nó não esteja presente na rede, a mesma deve tolerar estes problemas e continuar seu funcionamento de forma a alcançar o objetivo estabelecido. Estes sistemas são considerados tão desafiadores que seus conceitos são comumente estudados levando em consideração resultados decorrentes do Teorema de CAP.

2.2.1 Teorema de CAP

Este teorema, também conhecido como *Brewer's Theorem*, foi introduzido inicialmente em 1998 por Eric Brewer e provado como um teorema em 2002 por Seth Gilbert e Nancy Lynch.

O teorema determina que um sistema distribuído não pode ter Consistência (*Consistency*), Disponibilidade (*Availability*) e Tolerância a Partição (*Partition*) de forma simultânea (BREWER, 2000). A seguir cada um destes é explicado de forma rápida:

- **Consistência:** é a propriedade que garante que todos os nós em um sistema distribuído possuem pelo menos uma cópia atualizada dos dados;
- **Disponibilidade:** essa propriedade garante que um sistema está disponível, acessível para uso, e aceitando novas requisições e respondendo-as; com os dados sem nenhuma falha e sempre quando é requisitado;
- **Tolerância a Partição:** garante que caso um grupo de nós falhe, o sistema distribuído ainda continua a operar corretamente.

2.2.2 O Problema dos generais bizantinos

Nos primórdios da computação comercial, por volta de 1980, cientistas da computação começaram a examinar a confiabilidade e os computadores. Foi estabelecido que um sistema de computador confiável deveria ser capaz de lidar com a falta de um ou mais de seus componentes. Um componente com falha pode exibir um tipo de comportamento

negligenciado e problemático, ou seja, enviar informações conflitantes para diferentes partes do sistema. O problema de lidar com esse tipo de falta é expresso abstratamente como o Problema dos Generais Bizantinos, do artigo acadêmico de 1982 de Leslie Lamport, Robert Shostak e Marshall Pease (ALLEN, 2018).

O problema dos generais bizantinos é construído em torno do seguinte enredo no artigo de LAMPORT Leslie ;SHOSTAK (1982): várias divisões do exército bizantino estão acampados ao redor de uma cidade inimiga e cada uma dessas divisões é comandada por um general, onde estes podem se comunicar somente por meio de mensagens. Após observar o inimigo estes generais precisam chegar a um consenso em relação a um plano único de ataque, porém alguns destes generais podem ser traidores tentando fazer com que generais leais não cheguem a um acordo. Para que este acordo seja realizado, estes generais precisam de um algoritmo que garanta que todos os generais leais decidam sobre o mesmo plano de ação e que um pequeno número de generais traidores não façam com que os generais leais adotem um plano de ação ruim.

Na analogia, os generais são coletivamente conhecidos como processos, o general que inicia a comunicação com os outros é o processo de origem e as ordens enviadas aos outros processos são mensagens. Generais traidores são processos defeituosos, e generais leais são processos corretos. A ordem para recuar ou atacar é uma mensagem com um único bit de informação: um ou zero. Uma solução para este problema de acordo deve passar por três testes: rescisão, acordo e validade. Conforme aplicado ao problema dos generais bizantinos, Allen (2018) define que estes testes avaliam as seguintes características:

1. Uma solução deve garantir que todos os processos corretos cheguem a uma decisão em relação ao valor da ordem que eles receberam;

2. Todos os processos corretos devem decidir sobre o mesmo valor da ordem que receberam com os dados sem nenhuma falha e sempre quando é requisitado;

3. Se o processo de origem for um processo correto, todos os processos devem decidir sobre o valor que foi original dado pelo processo de origem.

Em seu artigo LAMPORT LESLIE ;SHOSTAK apresentam diferentes soluções que podem ser adotadas para solucionar este problema de consenso entre os generais. Estas abordagens descritas foram utilizadas na concepção da rede Blockchain como insumo para a criação dos diferentes protocolos de consenso utilizados para gerenciar esta rede distribuída.

2.3 Blockchain

Em 2008 um hacker anônimo (ou um grupo de hackers), utilizando o pseudônimo Satoshi Nakamoto, lançou a primeira moeda totalmente virtual e em pouco tempo divulgou um artigo (NAKAMOTO, 2008). Esse artigo denominado de *Bitcoin: A Peer-to-Peer Electronic Cash System* detalhava os métodos para o uso de uma rede peer-to-peer para gerar o que era descrito como "um sistema para transações eletrônicas sem abnegar a confiança".

Nakamoto combinou várias invenções anteriores, como b-money e HashCash para criar um sistema de caixa eletrônico completamente descentralizado que não dependia de uma autoridade central para emissão ou liquidação de moeda e validação de transações. A principal inovação foi usar um sistema de computação distribuída (chamado de algoritmo de "Prova-de-Trabalho") para conduzir uma "eleição" global a cada 10 minutos, permitindo a rede descentralizada chegar a um consenso sobre o estado das transações. Isso resolve elegantemente a questão do gasto duplo, em que uma única unidade monetária pode ser gasta duas vezes. Anteriormente, o problema do gasto duplo era uma fraqueza da moeda digital e era resolvido com a compensação de todas as transações por meio de uma central de compensação (ANTONOPOULOS, 2017).

O Bitcoin transformou totalmente os conceitos de bancos, dinheiro, processos de pagamentos entre outros; por meio da criação de um único registro digital acessível universalmente chamado de Blockchain. Foi atribuído esse nome de corrente porque mudanças só podem ser realizadas adicionando novas informações ao final, como em uma corrente de ferro, e cada nova adição a esta corrente contém um determinado números de transações.

2.3.1 Definição

A abstração para o entendimento de sistemas Blockchain é a noção de livro razão(Ledger), uma invenção desenvolvida pelos renascentistas italianos para suportar escrituração de dupla entrada, um precursor distante das criptomoedas modernas. Um livro contábil é apenas um registro indelével e unificado de transações que ocorrem entre várias partes. Um livro-razão estabelece quais transações aconteceram e a ordem em que essas transações aconteceram. Livros razão são públicos, acessíveis a todas as partes, e devem ser à prova de falsificação: nenhuma das partes pode adicionar, excluir ou modificar as entradas do livro, uma vez gravadas (HERLIHY, 2019).

Pode-se definir blockchain, em termos técnicos, como: "é um ledger distribuído de transações implementadas como dados agrupados em blocos que usam validação criptográfica para vincular os blocos. Cada bloco referencia e identifica o bloco anterior usando uma função de *hashing* que forma uma cadeia ininterrupta (isto é, blockchain)" (ALLEN, 2018).

De um ponto de vista empresarial, podemos caracterizar Blockchain como uma plataforma onde diferentes nós podem realizar troca de valores utilizando transações sem a necessidade de uma terceira parte autenticadora.

2.3.2 Tipos de Blockchain

Quando as pessoas começaram a entender o funcionamento do blockchain, começaram a usá-lo para outros fins: como armazenamento de dados para coisas de valor, identidades, acordos, direitos de propriedade e uma série de outras coisas (ALLEN, 2018). Cada uma dessas possíveis aplicações acaba demandando diferentes características e restrições para seu funcionamento, fazendo com que surgissem diferentes tipos e categorias de blockchain.

Os tópicos a seguir trarão uma pequena descrição sobre alguns dos diferentes tipos de blockchain.

2.3.2.1 Blockchains Públicos

Conforme definido pelos criadores originais da tecnologia, um blockchain público é aquele que tem como característica o fato de todos poderem acessar e realizar transações sem uma restrição para que novos nós sejam inseridos na rede; um blockchain onde as transações são incluídas se e somente se forem válidas; um blockchain onde todos podem contribuir para o processo de consenso (ALLEN, 2018).

No blockchain público, em vez de usar um servidor central, o blockchain é assegurado por verificação criptográfica por incentivos para os chamados mineradores (*miners*). Qualquer um pode ser um minerador para agregar e publicar essas transações. No blockchain público, como nenhum usuário é implicitamente confiável para verificar transações, todos os usuários seguem um algoritmo que verifica transações comprometendo recursos de software e hardware para resolver um problema por força bruta resolvendo um tipo de quebra-cabeça criptográfico (ALLEN, 2018).

2.3.2.2 Permissioned ledger

É um tipo de blockchain no qual os participantes da rede (nós) são conhecidos e já são confiáveis. Os registros autorizados não precisam usar um mecanismo de consenso distribuído. Em vez disso, um protocolo de acordo pode ser usado para manter uma versão compartilhada da verdade em relação ao estado dos registros no blockchain. Também não há exigência de que um *permissioned ledger* seja privado, pois pode ser um blockchain público, mas com controle de acesso regulado (BASHI, 2017).

2.3.2.3 Private Blockchain

Um blockchain totalmente privado é um blockchain no qual as permissões de gravação são mantidas centralizadas em uma organização. Permissões de leitura podem ser públicas ou restritas a uma extensão arbitrária. Aplicações prováveis incluem gerenciamento de banco de dados e auditoria interna para uma única empresa, de modo que a legibilidade pública pode não ser necessária em muitos casos, embora em outros casos a auditabilidade pública seja desejada (ALLEN, 2018).

2.3.3 Estrutura de um bloco

Um bloco é uma estrutura de dados que agrega transações para inclusão no livro razão público, o blockchain. O bloco é feito de um cabeçalho, contendo metadados, seguido por uma longa lista de transações que compõem a maior parte do seu tamanho. O cabeçalho do bloco é de 80 bytes, enquanto o tamanho médio das transações é de pelo menos 250 bytes e um bloco médio contém mais de 500 transações. Um bloco completo, com todas as transações, é, portanto cerca de 1.000 vezes maior que o cabeçalho do bloco (ANTONOPOULOS, 2017).

Um bloco é composto de uma seleção de transações organizadas de forma ordenada para que ocorra um encadeamento lógico. Esse bloco é composto por transações, e também contém uma referência ao bloco anterior. A estrutura do bloco varia de acordo com o tipo e design do Blockchain (BASHI, 2017).

A Tabela 1 apresenta a estrutura de um bloco de forma geral.

Tabela 1 – A estrutura de um bloco.

Tamanho	**Campo**	**Descrição**
4 bytes	Tamanho do Bloco	O tamanho do bloco, em bytes, após esse campo
80 bytes	Cabeçalho do Bloco	Os campos do cabeçalho do bloco
1-9 bytes	Contador da transação	Quantas transações seguem
Variável	Lista de Transações / Dados	As transações/dados gravados nesse bloco

2.3.4 Cabeçalho de um bloco

O cabeçalho do bloco consiste em três conjuntos de metadados do bloco. Primeiro, há uma referência a um hash do bloco anterior, o que conecta esse bloco ao bloco anterior na Blockchain. O segundo conjunto de metadados é composto pelos campos: a dificuldade(tempo necessário para o bloco ser gerado, onde este é atualizado a cada 2016 blocos), o registro de data e hora e o identificador único (contador), onde estes estão relacionadas à concorrência na mineração. A terceira parte dos metadados é a raiz da Árvore de Merkle, uma estrutura de dados usada para resumir eficientemente todas as transações no bloco (ANTONOPOULOS, 2017).

A Tabela 2 apresenta a estrutura de um cabeçalho de um bloco de forma geral.

Tabela 2 – Estrutura do cabeçalho de um bloco.

Tamanho	Campo	Descrição
4 bytes	Versão	A versão numérica para controle de atualizações
32 bytes	Hash do bloco anterior	Uma referência ao hash do bloco anterior (pai) na cadeia
32 bytes	Raiz da Árvore de Merkle	Um hash da raiz da Árvore de Merkle das transações deste bloco
4 bytes	Timestamp	O tempo aproximado de criação deste bloco
4 bytes	Dificuldade	A dificuldade do algoritmo de prova de trabalho para este bloco
4 bytes	Identificador Único	Um contador usado para o algoritmo de prova de trabalho

2.3.5 Block Header Hash e Block Height, identificadores de um bloco

O identificador primário de um bloco é seu hash criptográfico, uma impressão digital, feita pela aplicação do hash do cabeçalho do bloco duas vezes através do algoritmo SHA256. O hash de 32 bytes resultante é chamado de hash de bloco, mas é mais precisamente o hash de cabeçalho de bloco, porque somente o cabeçalho de bloco é usado para calculá-lo. O hash de bloco identifica um bloco de forma única e inequívoca e pode ser derivado independentemente por qualquer nó simplesmente fazendo hash no cabeçalho do bloco (ANTONOPOULOS, 2017).

Uma segunda maneira de identificar um bloco é por sua posição no blockchain, chamado altura do bloco, onde o primeiro bloco criado está na altura 0. Cada bloco subsequente adicionado “no topo” daquele primeiro bloco é uma posição “superior” no blockchain, como caixas empilhadas uma sobre a outra (ANTONOPOULOS, 2017).

Ao contrário do hash do bloco, a altura do bloco não é um identificador exclusivo. Embora um único bloco tenha sempre uma altura de bloco específica e invariante, o inverso não é verdadeiro, a altura do bloco nem sempre identifica um único bloco. Dois ou mais blocos podem ter a mesma altura de bloco, competindo pela mesma posição no blockchain.

2.3.6 Consenso no blockchain

O consenso é um processo de concordância entre nós que não possuem confiança mútua em um estado final de dados. Para alcançar consenso, diferentes algoritmos podem ser usados. É fácil chegar a um acordo entre dois nós (por exemplo, em sistemas cliente-servidor), mas quando vários nós participam de um sistema distribuído e precisam concordar em um único valor, torna-se muito difícil chegar a um consenso. Esse conceito de alcançar consenso entre vários nós é conhecido como consenso distribuído (BASHI, 2017).

Um mecanismo de consenso é um conjunto de etapas tomadas por todos, ou a maioria dos nós, para chegar a um acordo sobre um estado ou valor proposto. Por mais de

três décadas este conceito tem sido pesquisado por cientistas da computação na indústria e na academia. Mecanismos de consenso recentemente entraram no centro das atenções e ganharam muita popularidade com o advento do bitcoin e blockchain (BASHI, 2017).

Cada blockchain tem seus próprios algoritmos para criar um acordo dentro de sua rede nas entradas adicionadas, onde o processo de entrada pode ser visualizado na Figura 10. Existem muitos modelos diferentes para criar consenso porque cada blockchain está criando diferentes tipos de entradas. Alguns dos blockchains são de valor comercial, outros armazenam dados e outras estão protegendo sistemas e contratos (LAURENCE, 2019).

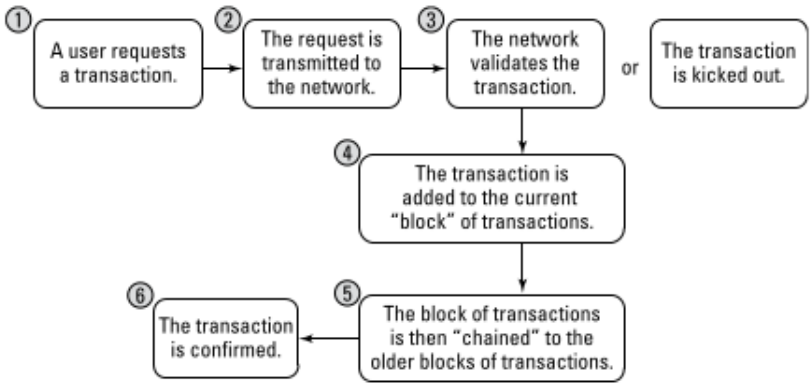


Figura 10 – Funcionamento do Blockchain - Imagem retirada de (LAURENCE, 2019).

Os mecanismos de consenso vêm da teoria dos jogos. Os sistemas devem ser projetados de tal forma que os nós obtenham o maior benefício se eles seguirem as regras. Um dos aspectos para garantir que os nós se comportem honestamente é recompensar o comportamento honesto e punir por atividades fraudulentas. No entanto em um blockchain público como o do Bitcoin, um nó pode ter muitas identidades públicas diferentes e anônimas. Fica muito difícil punir essas identidades porque elas têm a opção de evitar essa punição criando novas identidades para si mesmas. Por outro lado, recompensá-los funciona muito bem, porque mesmo que alguém tenha múltiplas identidades, eles podem colher os frutos que lhes são dados (PANDA, 2018).

O objetivo do consenso é também garantir que a rede seja robusta o suficiente para sustentar vários tipos de ataques. Independentemente dos tipos de algoritmos de consenso que um pode escolher, dependendo do caso de uso, ele deve cair no molde de um consenso tolerante a falhas bizantinas para ser aceito (PANDA, 2018).

2.3.6.1 Protocolos de consenso

A escolha do protocolo de consenso afeta a segurança e a escalabilidade. Uma vez que um novo bloco é gerado por um *miner*, o *miner* propaga o bloco para seus pares conectados na rede blockchain. No entanto, os *miners* podem encontrar novos blocos

competidores diferentes e resolver isso usando os mecanismos de consenso do blockchain (STAPLES, 2019).

A abordagem geral típica é chamada de consenso Nakamoto. Essa abordagem depende dos participantes que selecionam como autoritativa a maior cadeia de blocos que observaram em cada ponto no tempo. No Bitcoin, novos blocos são gerados através de um mecanismo de prova de trabalho (*Proof Of Work*). A prova de trabalho usa um quebra-cabeça criptográfico que é fácil de verificar, mas é difícil de resolver e toma um tempo efetivamente aleatório. Os mineradores de Bitcoin competem para resolver esse quebra-cabeça para cada bloco, usando grandes quantidades de computações (e, portanto, de eletricidade) para aumentar suas chances de vencer a competição pelo bloco. O investimento requerido pelas mineradoras para isso serve para alinhar seus incentivos com o bom funcionamento do sistema como um todo. Existem vários mecanismos de prova de trabalho, como o Ethash 11 usado pelo Ethereum e o Hashcash 12 usado pelo Bitcoin (STAPLES, 2019).

A prova de participação (*Proof of stake*) é um mecanismo alternativo para o consenso Nakamoto, que seleciona o próximo nó de mineração com base no controle da moeda digital nativa da rede blockchain. Por exemplo, os mineradores em Peercoin 14 precisam provar a propriedade de uma certa quantia de moeda de Peercoin para os blocos de mineração. Assim, a prova de participação naturalmente alinha os incentivos dos detentores de moeda digital no blockchain com o bom funcionamento do blockchain. Existem vários protocolos de prova de participação, por ex. Tendermint 15 usado em Eris e Casper 16 para Ethereum. Comparado com a prova de trabalho, a prova de participação é mais eficiente em termos de custo, porque muito menos poder computacional é usado na mineração e a latência também é menor (STAPLES, 2019).

PBFT é o acrônimo para o algoritmo de Tolerância de Falha Bizantina Prática e é um dos muitos algoritmos de consenso que se pode considerar para uso em um blockchain, embora seja um algoritmo que não é usado para gerar recompensas de mineração. O funcionamento interno do PBFT, em um alto nível, é transmitido para todos os nós participantes que possuem suas próprias réplicas ou estados internos. Quando os nós recebem uma solicitação, eles executam o cálculo com base em seus estados internos. O resultado da computação é então compartilhado com todos os outros nós no sistema. Então, todo nó está ciente do que outros nós estão computando. Considerando seus próprios resultados de computação, juntamente com os recebidos de outros nós, eles tomam uma decisão e se comprometem com um valor final, que é novamente compartilhado entre os nós. Neste momento, cada nó está ciente da decisão final de todos os outros nós. Em seguida, todos respondem com suas decisões finais e, com base na maioria, o consenso final é alcançado. O PBFT pode ser eficiente em comparação com outros algoritmos de consenso, com base no esforço necessário. No entanto, o anonimato no sistema pode ser comprometido devido

à maneira como esse algoritmo é projetado (PANDA, 2018).

2.3.7 Smart Contracts

Um cientista de computação e especialista em criptografia chamado Nick Szabo é creditado por desenvolver o conceito de um contrato inteligente. Ele definiu um contrato inteligente como "um conjunto de promessas, especificadas em formato digital, incluindo protocolos nos quais as partes cumprem essas promessas". (SZABO, 1996)

Assim, um contrato inteligente é um algoritmo computadorizado, que executa os termos do contrato. No entanto, essa definição não diferencia contratos inteligentes de algumas construções contratuais já bem conhecidas que implementam o desempenho automatizado (desempenho que não depende de intervenções externas para que um conjunto de atividades sejam executadas) em seu funcionamento como máquinas de venda automática. Máquinas de venda automática são definidas como máquinas automáticas independentes que distribuem bens ou prestam serviços quando as moedas são inseridas ou o pagamento por, digamos, um cartão de crédito é feito. Máquinas de venda automática são programadas com certas regras e executadas por essas regras. (ALLEN, 2018)

Se não houver uma diferença principal entre as máquinas de venda automática e os contratos inteligentes, os contratos inteligentes são tão antigos quanto a própria lei romana. No primeiro século AC, um engenheiro e matemático grego, Heron de Alexandria, documentou a primeira máquina de venda automática. Sua máquina aceitava uma moeda de dracma e depois distribuia água benta em templos da região. Portanto, os contratos inteligentes, por si só, não são novos (ALLEN, 2018).

2.4 Ethereum

A intenção do Ethereum é fundir e melhorar os conceitos de scripting, altcoins e meta-protocolos on-chain, e permitir que os desenvolvedores criem aplicativos arbitrários baseados em consenso que tenham escalabilidade, padronização, funcionalidade completa, facilidade de desenvolvimento e interoperabilidade oferecidas por esses diferentes paradigmas, simultaneamente. A Ethereum faz isso construindo o que é essencialmente a última camada fundacional abstrata: um blockchain com uma linguagem de programação Turing-completa integrada, permitindo que qualquer um escreva contratos inteligentes e aplicativos descentralizados onde possam criar suas próprias regras arbitrárias de propriedade, formatos de transação e funções de transição de estado (BUTERIN, 2013).

No Ethereum as chamadas "mensagens" se assemelham as transações no Bitcoin, porém com algumas diferenças. Primeiramente, uma mensagem Ethereum pode ser criada por uma entidade externa (usuário) ou por um contrato; segundo, existe uma opção explícita para as mensagens conterem dados; a terceira diferença é que o receptor de

uma mensagem Ethereum tem a opção de responder, ou seja, as mensagens ethereunm englobam o conceito de funções (BUTERIN, 2013).

Um contrato inteligente no Ethereum é uma caixa protegida criptograficamente que contém lógica e valor. A lógica tem condições específicas que precisam ser atendidas para liberar o valor. É a adição de lógica e estado que torna o Ethereum uma plataforma mais poderosa do que a que está disponível com outras criptomoedas, incluindo o Bitcoin (e sua funcionalidade de script). Um exemplo de um contrato inteligente para uma empresa de seguros poderia ser utilizado para pagar automaticamente para a entidade segurada com base em um feed de dados que mostrassem que um evento pagável ocorreu.(ALLEN, 2018)

Contratos inteligentes permitem aplicativos descentralizados (Dapps). Por exemplo, quando interagimos com um aplicativo de terceiros em nosso *smartphone*, o aplicativo se comunicará com servidores e serviços centralizados. Um Dapp pode ser exatamente o mesmo em termos da interface do usuário, mas os serviços de *back-end* são substituídos por contratos inteligentes que são executados na rede Ethereum descentralizada (ALLEN, 2018).

O mecanismo de contrato descrito acima permite que qualquer pessoa construa o que é essencialmente um aplicativo de linha de comando executado em uma máquina virtual que é executada por consenso em toda a rede, permitindo modificar um estado globalmente acessível como seu "disco rígido". No entanto, para a maioria das pessoas, a interface de linha de comando que é o mecanismo de envio de transações não é suficientemente amigável ao usuário para tornar a descentralização uma alternativa convencional atraente. Para este fim, uma aplicação descentralizada completa deve consistir em componentes lógicos de negócios de baixo nível, implementados inteiramente na plataforma Ethereum, usando uma combinação de Ethereum e outros sistemas (por exemplo, uma camada de mensagens P2P), e componentes gráficos de interface de usuário de alto nível. O design do cliente Ethereum serve como um navegador da *Web*, mas inclui suporte para um objeto de API JavaScript "eth", que páginas especificas da *Web* visualizadas no cliente poderão usar para interagir com o blockchain Ethereum. Do ponto de vista da *web* "tradicional", essas páginas são inteiramente de conteúdo estático, uma vez que o blockchain e outros protocolos descentralizados servirão como um substituto completo do servidor com a finalidade de manipular solicitações iniciadas pelo usuário (BUTERIN, 2013).

2.4.1 Aplicações Descentralizadas (Dapp)

Um aplicativo descentralizado (Dapp) é como qualquer outro aplicativo moderno arquitetado pela Web, geralmente consistindo em uma interface com o usuário (UI) cuja funcionalidade é suportada pelos serviços de *back-end* (leitura e gravação em armazenamento persistente, processamento, lógica complexa). Esses serviços de *back-end* geral-

mente utilizam a plataforma Ethereum e, em particular, os contratos inteligentes que são implantados para ela. Em retrospecto, pode-se dizer que de certa forma o primeiro Dapp conhecido foi o Bitcoin, porém este não foi implementado na plataforma Ethereum (ALLEN, 2018).

A Figura 11 representa um exemplo da estrutura de um aplicativo descentralizado.

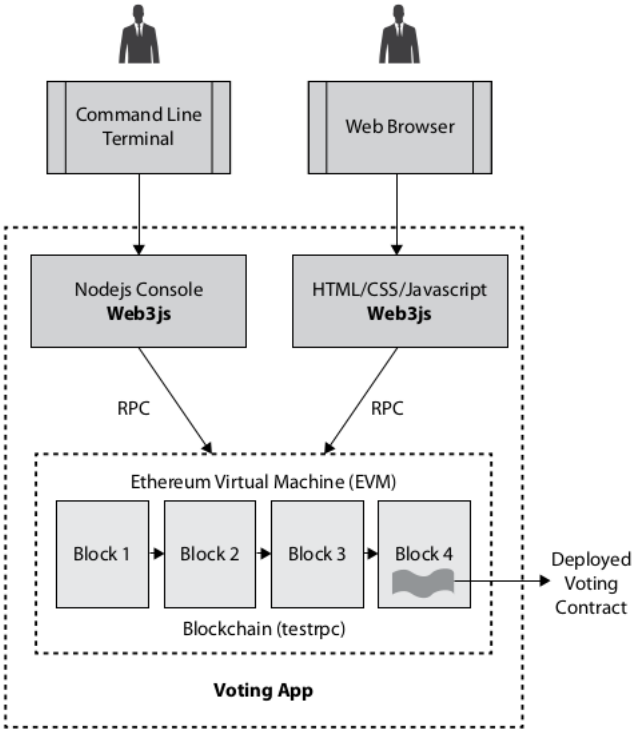


Figura 11 – Aplicação Descentralizada Ethereum - Imagem retirada de (ALLEN, 2018).

2.5 Carteira de Criptomoeda

Uma carteira digital para criptomoedas é um programa de software que contém chaves públicas e privadas e funciona com sucesso em diferentes blocos de blocos que permitem aos usuários trocar moedas entre si e controlar o saldo de sua moeda. Uma carteira digital pode armazenar, enviar e receber moedas diferentes. As criptomoedas não são armazenadas como dinheiro físico na carteira. Cada transação é registrada e armazenada no blockchain. Enviar um Bitcoin ou alguma outra moeda para um usuário significaria enviar sua própria chave pública. Se o usuário deve receber o pagamento, sua chave privada deve estar de acordo com a chave pública do remetente. Não há troca real de moedas. A transação é concluída com um registro no blockchain e uma mudança na carteira digital do usuário. (ROSIC, 2018)

As carteiras de criptomoeda podem ser divididas em duas categorias principais: carteiras frias e carteiras quentes. A diferença entre dois deles é que para carteiras quen-

tes é necessária uma conexão à Internet e para carteiras frias não. Os usuários de hot wallet costumam usá-los para comprar algo na internet e manter uma pequena quantia de dinheiro para esse fim, enquanto cold wallet é como um cofre no banco para armazenar diferentes tipos de valores digitais. O melhor é ter as duas carteiras principalmente por motivos de segurança. As carteiras quentes e frias podem ser divididas em várias categorias: carteiras online, carteiras de hardware, carteiras móveis, carteiras de papel e carteiras de mesa. As carteiras online, mobile, de mesa e multisignature pertencem a uma categoria de carteira quente e hardware, o papel pertence a uma categoria de carteira fria. Dependendo da escolha da carteira digital, cada uma tem seu próprio nível de segurança para garantir a proteção da chave privada. (JOKIć et al., 2019)

2.6 Código de Trânsito Brasileiro

O Código de Trânsito Brasileiro(CTB) é um documento legal que foi instituído em 1997 pelo presidente da República por meio da Lei n 9.503, substituindo o Código Nacional de Trânsito de 1966 que era responsável por regular as normas de trânsito no Brasil até então. As regras de trânsito definidas no CTB são atualizadas todos os anos pelos poderes Executivo e Legislativo, por meio da publicação de leis que atualizam os artigos do CTB.

O CTB assegura que o trânsito é um direito de todos e atribui aos órgãos e entidades componentes do Sistema Nacional De Trânsito (SNT) a responsabilidade de adotar medidas destinadas a assegurar esse direito. Considera-se trânsito: “a utilização das vias por pessoas, veículos e animais, isolados ou em grupos, conduzidos ou não, para fins de circulação, parada, estacionamento e operação de carga ou descarga” (PRESIDêNCIA DA REPúBLICA, 1997).

O Sistema Nacional de Trânsito é definido no CTB pelo artigo 5 como: “... conjunto de órgãos e entidades da União, dos Estados, do Distrito Federal e dos Municípios que tem por finalidade o exercício das atividades de planejamento, administração, normatização, pesquisa, registro e licenciamento de veículos, formação, habilitação e reciclagem de condutores, educação, engenharia, operação do sistema viário, policiamento, fiscalização, julgamento de infrações e de recursos e aplicação de penalidades”.

Segundo o artigo 7 do CTB, o Sistema Nacional de Trânsito é composto dos seguintes órgãos e entidades:

> “Art. 7º- Compõem o Sistema Nacional de Trânsito os seguintes órgãos e entidades:
>
> I. o Conselho Nacional de Trânsito - CONTRAN, coordenador do Sistema e órgão máximo normativo e consultivo;

II. os Conselhos Estaduais de Trânsito - CETRAN e o Conselho de Trânsito do Distrito Federal - CONTRANDIFE, órgãos normativos, consultivos e coordenadores;;

III. os órgãos e entidades executivos de trânsito da União, dos Estados, do Distrito Federal e dos Municípios;

IV. os órgãos e entidades executivos rodoviários da União, dos Estados, do Distrito Federal e dos Municípios;

V. a Polícia Rodoviária Federal;

VI. as Polícias Militares dos Estados e do Distrito Federal; e

VII. as Juntas Administrativas de Recursos de Infrações - JARI."

2.6.1 Infração de Trânsito

O CTB define no Capítulo 15, Artigo 161 infração como:

"... a inobservância de qualquer preceito deste Código, da legislação complementar ou das resoluções do CONTRAN, sendo o infrator sujeito às penalidades e medidas administrativas indicadas em cada artigo, além das punições previstas no Capítulo XIX."

A multa é uma das penalidades aplicadas aos motoristas que cometeram uma ou mais das infrações elencadas no CTB, onde a multa é uma penalidade relacionada a todos os outros tipos de infração. A lista das penalidades aplicadas está elencada no artigo 256, sendo estas:

I. advertência por escrito;

II. **multa**;

III. suspensão do direito de dirigir;

IV. apreensão do veículo; (Revogado pela Lei nº 13.281, de 2016)

V. cassação da Carteira Nacional de Habilitação;

VI. cassação da Permissão para Dirigir;

VII. frequência obrigatória em curso de reciclagem."

2.6.1.1 Tipos de Infração

O CTB categoriza os tipos de infrações de trânsito em quatro diferentes categorias: leves, médias, graves, gravíssimas. Cada categoria possui um valor diferente de

penalização, onde as infrações de natureza gravíssima podem ter seu custo aumentado pelos fatores multiplicadores previstos no parágrafo 2 do artigo 258.

Os valores definidos no artigo 258 para cada natureza de infração são apresentados na Tabela 3:

Tabela 3 – Valor da penalização de cada categoria das infrações previstas no artigo 258 do CTB em Junho de 2019.

Categoria da Infração	**Valor**
Leve	R$ 88,38
Média	R$ 130,16
Grave	R$ 195,23
Gravíssima	R$ 293,47

Além da penalização monetária aplicada, uma pontuação também é computada à Carteira Nacional de Trânsito, CNH, do condutor identificado como condutor no ato da infração de acordo com a natureza da mesma. A pontuação atribuída ao infrator é regulamentada pelo artigo 259 do CTB, que apresenta a seguinte redação:

> "Art. 259. A cada infração cometida são computados os seguintes números de pontos:
>
> I. gravíssima - sete pontos;
>
> II. grave - cinco pontos;
>
> III. média - quatro pontos;
>
> IV. leve - três pontos."

2.6.1.2 Auto de Infração

Constatada a infração, será lavrado o Auto de Infração, que deverá conter os requisitos mínimos definidos pelo artigo 280 do CTB (PRESIDêNCIA DA REPúBLICA, 1997):

> " Art. 280. Ocorrendo infração prevista na legislação de trânsito, lavrar-se-á auto de infração, do qual constará:
>
> I. tipificação da infração;
>
> II. local, data e hora do cometimento da infração;
>
> III. caracteres da placa de identificação do veículo, sua marca e espécie, e outros elementos julgados necessários à sua identificação;
>
> IV. o prontuário do condutor, sempre que possível;

V. identificação do órgão ou entidade e da autoridade ou agente autuador ou equipamento que comprovar a infração;

VI. assinatura do infrator, sempre que possível, valendo esta como notificação do cometimento da infração."

2.6.2 Registro de Infrações de Trânsito

O registro das infrações de trânsito é feito pelo órgão ou entidade responsável pela jurisdição onde a infração foi cometida, sendo a mesma podendo ser salva em um sistema local do órgão caso a infração seja registrada na mesma unidade federativa que o veículo está licenciado ou no Registro Nacional de Infrações de Trânsito (RENAINF) caso o veículo não seja licenciado no estado onde ocorreu a infração. Por exemplo, no Distrito Federal os órgãos e entidades responsáveis por fazer fiscalizações e registrar infrações são: Departamento de Trânsito Do Distrito Federal (Detran-DF), Departamento de Estradas de Rodagem (DER) e Polícia Militar (PMDF).

O RENAINF - Registro Nacional de Infrações de Trânsito é um sistema coordenado pelo Departamento Nacional de Trânsito (DENATRAN) que registra as infrações à legislação de trânsito cometidas em unidade federada diversa daquela onde o veículo estiver registrado e licenciado, bem como permite o registro das infrações impostas pela Polícia Rodoviária Federal (PRF), Agência Nacional de Transportes Terrestres (ANTT) e o Departamento Nacional de Infraestrutura de Transportes (DNIT), independente da vinculação de registro do veículo. Esse sistema possibilita que o órgão autuador tenha os dados necessários para notificar o proprietário do veículo sobre a infração cometida e sobre a respectiva penalidade, e vincular os débitos existentes no DETRAN de registro do veículo.(FAZENDA E PLANEJAMENTO DE SãO PAULO, 2018)

3 Proposta de Trabalho

A proposta de trabalho consiste no desenvolvimento de uma aplicação descentralizada(Dapp), por meio do Ethereum, para o registro de infrações de trânsito utilizando-se da tecnologia *blockchain* para garantir a integridade das informações registradas.

Será implementado um blockchain do tipo público, onde todos podem contribuir para o processo de consenso e validação dos blocos adicionados, possibilitando que qualquer pessoa interessada possa contribuir para a rede de registros se tornando um nó para validar as transações. Para mais informações em relação a este tipo de blockchain, veja a Seção 2.3.2

Para auxiliar nos testes e uso da aplicação descentralizada durante o desenvolvimento, serão utilizadas duas ferramentas: Ganache e o Metamask.

O Ganache [1] permite ativar e gerenciar rapidamente um blockchain Ethereum pessoal para executar testes, executar comandos e inspecionar o estado do mesmo enquanto controla como a cadeia de blocos opera tanto pelo terminal de operações do computador como por uma interface gráfica. Dessa maneira pode-se testar os contratos inteligentes desenvolvidos sem o custo da rede Ethereum principal.

O Metamask [2] é uma carteira virtual e pode ser adicionada ao navegador como um plugin, que serve como uma ponte para conectar-se à rede desejada e injetar uma instância web3 no código. O Metamask possibilita selecionar e navegar entre diferentes redes (como uma rede local ou a principal Ethereum), assim como disponibiliza uma interface para gerenciar uma carteira digital com diferentes contas. Esta funcionalidade possibilitará que os diferentes níveis de restrições de usuários, como o usuário normal e o administrador, sejam testados durante todo o processo de desenvolvimento de forma prática.

Para a realização do trabalho, algumas premissas serão tomadas como base para seu desenvolvimento. As premissas são:

I. O sistema funcionará como o RENAINF(vide Seção 2.6.2), ou seja o sistema único para registro de infrações em todo o território nacional, onde todas infrações serão registradas nesta plataforma, mesmo caso o infrator tenha o carro licenciado na unidade federativa onde ocorreu a violação da norma;

II. Cada órgão ou entidade do sistema nacional de trânsito terá sua representação no sistema por meio de uma conta de administrador;

[1] https://www.trufflesuite.com/docs/ganache/quickstart
[2] https://metamask.io/

III. Serão utilizados bancos de dados auxiliares para simular as bases de dados de condutores habilitados e veículos registrados, de forma que essas informações possam ser obtidas para o registro da infração;

IV. Cada condutor habilitado terá seu próprio conjunto de chaves público/privada, onde a chave pública será salva junto ao registro do condutor no banco de dados auxiliar responsável por esta informação.

3.1 Aplicação proposta

Uma aplicação descentralizada será criada para tornar o processo de registro infrações de trânsito transparente e descentralizado. A biblioteca React escrita em javascript será usado para implementar o *front-end* e os contratos inteligentes da plataforma Ethereum serão usados no *back-end*.

O ReactJs é uma biblioteca javaScript usada para criar interfaces de usuário. A principal característica é que podemos criar nossos aplicativos de maneira modular, o que ajuda a reduzir a repetição de códigos e facilita a depuração. Ele também permite modificar os elementos HTML dinamicamente, facilitando a criação de páginas interativas em tempo real.

Os contratos inteligentes escritos em `solidity` para a rede Ethereum no back-end da aplicação torna o Dapp descentralizado e transparente. A linguagem `solidity` também é totalmente suportada pelo Ethereum e tem uma grande comunidade ao seu redor, o que auxilia no aprendizado e na solução de possíveis empecilhos ao longo do desenvolvimento do projeto.

3.1.1 Arquitetura Proposta

A arquitetura da aplicação será no estilo cliente servidor. O Dapp conterá duas partes principais: o contrato, sendo executado na rede Ethereum, e a aplicação cliente sendo executado localmente no computador do usuário.

Um exemplo dessa arquitetura pode ser vista na Figura 12.

3.1.2 Funcionalidades

De forma a contemplar o máximo de situações contempladas no sistema RENAINF e respeitando a regulação presente no CTB, o sistema conterá as seguintes funcionalidades:

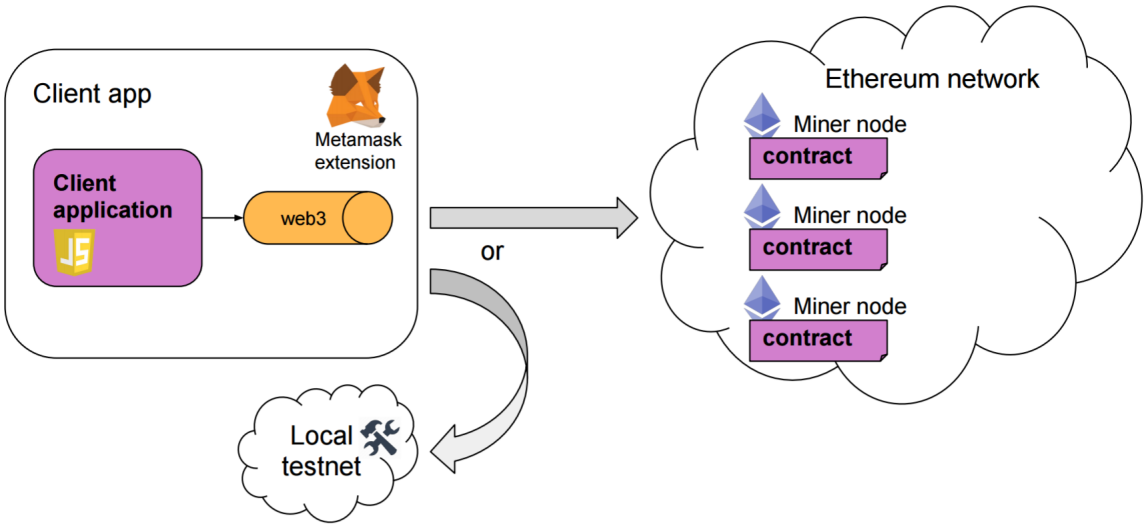


Figura 12 – Representação das partes - Imagem retirada de Dapp Architecture.

3.1.2.1 Registro de Infração

No registro de uma infração, os órgãos ou entidades registrados como administradores vão inserir no sistema as informações relacionadas ao registro de infração. Nesta plataforma inicialmente não serão registrados todos os campos elencados no CTB (Seção 2.6.1.2), as informações que serão cadastradas no sistema para o registro da infração de forma a contemplar as informações nesses campos são:

- Placa do veículo
- Categoria da Infração
- Data/Hora da Infração
- Pontos da Infração
- Observações
- Chave do administrador responsável pelo registro
- Chave do motorista responsável pelo veículo
- Preço para pagamento da infração (valor em ethers)

Após confirmar as informações preenchidas, a transação será inserida no blockchain e validada por outros nós. Após este processo a infração estará disponível para que sejam realizadas as outras funções como consulta, realização de recursos e pagamento da mesma.

3.1.2.2 Transferência de Infração

Nesta funcionalidade, o usuário indicado como autor da infração registrada poderá transferir para outro usuário a autoria da infração de trânsito.

Ao visualizar as informações da infração, o usuário(A) terá uma opção disponível para inserir a chave de outro usuário (B), e realizar a indicação de autoria. Este usuário (B) terá em sua página uma indicação dessa solicitação e terá disponível as opções: aceitar ou recusar. Caso o usuário (B) aceite a transferência assinando a transação com sua chave, a transação indicando a transferência será realizada de forma automática sem a necessidade da interferência de um usuário administrador (C).

3.1.2.3 Busca de Infrações

A busca de infrações consistirá em dois módulos diferentes: busca realizada pelo administrador e busca do usuário normal.

O usuário registrado como administrador poderá consultar os registros de diferentes formas, como: infrações de um usuário específico, ou por um veículo.

Já o usuário normal terá em sua página de acesso todas as infrações que foram registradas como ele sendo o autor da infração, não sendo possível visualizar as infrações realizadas por outros usuários.

3.1.2.4 Pagamento Infração

O pagamento da infração poderá ser realizada pelo usuário indicado como responsável pela infração. Para este pagamento serão utilizadas as moedas virtuais da rede Ethereum, de acordo com o valor indicado na infração acrescido das taxas necessárias para a realização da transação(GAS). O pagamento de uma infração corresponde à ação de transferir os valores referentes ao pagamento da infração para a conta da autoridade que registrou a infração.

3.1.2.5 Recurso / Cancelamento de uma infração

Ao visualizar a infração o usuário (A) indicado como responsável também poderá entrar com recurso em relação a infração registrada. Esta condição se aplica a casos onde o usuário não concorda com a infração registrada ou possui uma explicação plausível que pode acarretar na anulação da mesma.

Para realizar esta ação o usuário (A) deverá clicar na opção disponível, preencher as informações solicitadas e assinar a solicitação usando sua chave. Essa solicitação irá para análise por parte de um usuário(B) administrador com perfil indicado como um Junta de Trânsito, e o mesmo poderá recusar ou aceitar a solicitação realizada.

Caso o usuário (B) administrador aceite o recurso realizado, uma transação será inserida no blockchain anulando a infração registrada anteriormente e removendo a responsabilidade do usuário (A) em relação a mesma removendo as penalidades aplicadas anteriormente.

3.1.3 Minerar moedas

Para minerar moedas Ethereum é necessário ter um computador com alta capacidade de processamento. O papel destes mineradores é encontrar uma sequência que torne um bloco de transações compatível com o bloco anterior. Para isso, o computador precisa efetuar milhares de cálculos por segundo para encontrar a combinação perfeita, por isso que eles precisam ser extremamente potentes. Ao encontrar a sequência compatível, o minerador recebe uma recompensa para cada bloco que ele minerar. Essa recompensa foi criada com a intenção de pagar as pessoas que emprestam poder computacional para manter a rede funcionando.

De forma a incentivar a inserção do maior número de nós possíveis na rede atuando como mineradores, tornando a confiabilidade da rede ainda maior, o sistema englobará uma funcionalidade que permitirá que o pagamento das infrações registradas no blockchain seja feito utilizando essa recompensa monetária obtida por meio da mineração. Este incentivo beneficiará principalmente pessoas jurídicas que possuem uma grande frota de veículos automotores e serviços computacionais a disposição, como empresas locadoras de veículos e empresas de transporte urbano, pois estes poderão pagar as infrações realizadas por sua frota com um certo desconto em relação ao valor normal ao ajudar na manutenção da rede.

4 Metodologia

4.1 Metodologia de Desenvolvimento

Para o desenvolvimento do trabalho será adotada uma metodologia híbrida, contendo diferentes ferramentas de abordagens de desenvolvimento de software. Essa personalização tem o objetivo de otimizar o desenvolvimento das tarefas no contexto em que esse trabalho é desenvolvido, tendo como uma das motivações principais para realizar essa personalização o fato do trabalho não ser construído em grupo.

Será utilizada a metodologia KanBan, com algumas personalizações, para gerenciar a execução de atividades. Os cartões utilizados neste quadro estarão representados no formato de *User Story*, comumente utilizada na metodologia SCRUM, com seu conteúdo sendo composto por: uma descrição no formato da sentença **Eu como <quem>, quero <o que>, para <ação>**, as *tasks* a serem desenvolvidas, o tempo limite para sua realização e a categoria da mesma.

O quadro será composto por 4 diferentes colunas: A fazer, Fazendo, Aguardando Validação e Finalizados. A imagem a seguir exemplifica o fluxo de desenvolvimento no quadro *KanBan* que será utilizado no trabalho.

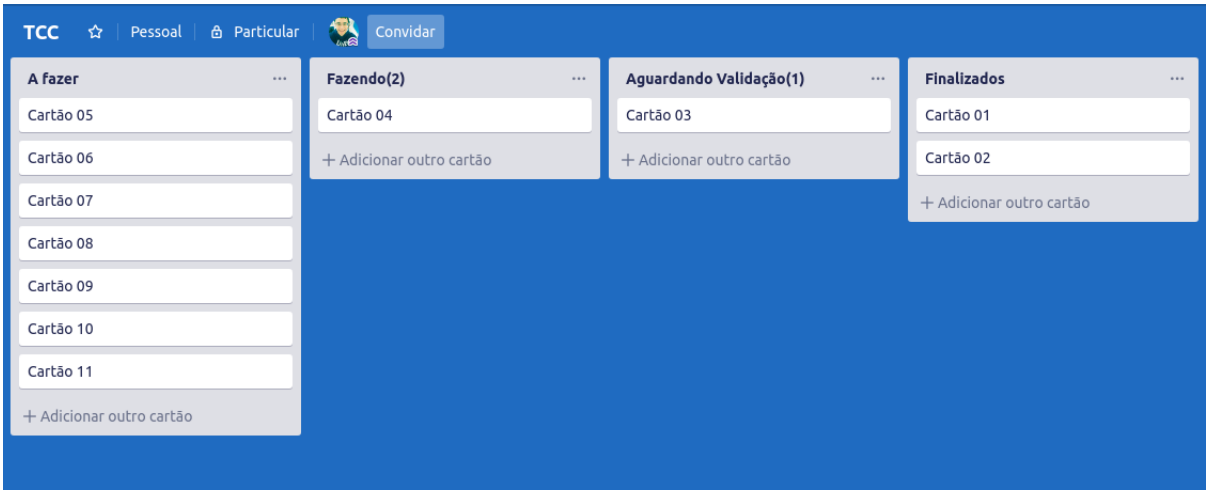


Figura 13 – KanBan representado na ferramenta Trello

A coluna **A Fazer** funcionará como um *Backlog* das funcionalidades que serão desenvolvidas ao decorrer do trabalho (possíveis *bugs* ou melhorias que surjam ao decorrer do projeto também serão adicionadas à essa coluna), onde os cartões posicionados acima são os considerados de maior prioridade para serem desenvolvidos. A coluna **Fazendo**

será responsável por indicar o cartão que está que está em processo de desenvolvimento. Já a **Aguardando Validação** indicará o cartão que está passando por um processo de validação, como por exemplo a realização de testes com usuários reais para validar o funcionamento implementado em relação ao descrito no cartão. E a coluna **Finalizados** conterá o histórico das funcionalidades já implementadas e que foram validadas de acordo com os testes planejados.

Para melhor monitorar o andamento do trabalho, o planejamento dos cartões será feito por meio de iterações semanais. Neste planejamento semanal poderão ocorrer mudanças nas prioridades dos cartões presentes no quadro *KanBan*, inserção de novos cartões na fila **A fazer**, remoção do quadro cartões que são considerados não necessários ou que precisem ser desmembrados em outros cartões pela complexidade da atividade.

4.2 Políticas para o desenvolvimento do Projeto

Toda a aplicação terá sua construção feita na plataforma GitHub em um repositório fechado. As issues do GitHub serão utilizadas como os cartões do KanBan para o cadastro das informações necessárias, e serão gerenciados de forma visual por meio da ferramenta ZenHub.

A cada adição de funcionalidade ao sistema um novo *commit* será feito para um melhor versionamento e controle do que será desenvolvido, e cada *commit* terá seu texto no padrão **#IdDaIssue Descrição**, como no seguinte caso: "#1 - Creating user structure".

O repositório do projeto contará com duas branches principais para o desenvolvimento, sendo elas: *master* e *devel*. A branch *master* conterá a versão estável do projeto, tendo seu conteúdo proveniente da branch de desenvolvimento (*devel*), que será utilizada para o desenvolvimento do projeto, após a validação do que foi desenvolvido. Por decisões de projeto, nenhum *commit* poderá ser feito diretamente na *branch master*, onde para inserir novos arquivos deve ser realizado um *merge/pull request* tendo como origem das informações a *branch devel*.

5 Resultados

5.1 Solução Desenvolvida

5.1.1 Tecnologias e ferramentas utilizadas

Para a construção da proposta de solução final foram utilizadas algumas ferramentas, tecnologias e bibliotecas ao longo do processo de desenvolvimento. A seguir estas estão elencadas junto com a motivação de sua utilização dentro da solução:

- ReactJS: Biblioteca javascript utilizada para criar a interface dos diferentes módulos da solução.
- Material UI: Biblioteca de componentes ReactJS para um desenvolvimento ágil e fácil, sendo utilizado para criar um visual mais agradável para a aplicação.
- Web3.Js: Coleção de bibliotecas que possibilitam a interação com nós ethereum remotos, utilizando conexões HTTP ou ICP. Possibilitando assim que os clientes possam se comunicar com o contrato inteligente desenvolvido.
- Metamask: Extensão de navegador para acesso a aplicativos distribuídos(Dapps), onde a mesma injeta a API Ethereum do Web3 no contexto javascript dos sites. Também permite que os usuários criem e gerenciem suas próprias identidades e transações, trazendo uma interface segura para revisar transações antes de aceitá-las ou rejeitá-las.
- Docker e Docker-compose: Containers utilizados para facilitar o gerenciamento de dependências da aplicação assim como orquestrar facilmente os ambientes de desenvolvimento e produção das mesmas.
- TESTNET Ropsten (ETH): Rede ethereum que executa o mesmo protocolo que o Ethereum(main net) e é usada para fins de teste. O contrato inteligente da solução está registrado nesta e rede e todas as transações ocorrem em interações dentro dela.
- Ropsten Ethereum Faucet: Ferramenta utilizada para obter ethers gratuitos da rede de testes Ropstas para as contas utilizadas na solução;
- Digital Ocean: Serviço de hospedagem utilizado para hospedar os módulos(Dapps) desenvolvidos em um servidor, possibilitando que as aplicações possam ser acessadas sem a necessidade de executá-las localmente.

- Solidity: Linguagem de programação de alto nível por meio da qual é possível criar aplicações descentralizadas, em especial os smart contracts, na Ethereum.
- Truffle: Um ambiente de desenvolvimento, framework de testes e pipeline de ativos para blockchains usando a Ethereum Virtual Machine (EVM). Tendo sido utilizado principalmente na fase inicial de construção do contrato inteligente, possibilitando o desenvolvimento do mesmo mais rapidamente.
- Ganache: Blockchain Ethereum pessoal utilizado executar testes, executar comandos e inspecionar o estado enquanto controla como a cadeia de blocos opera localmente.

5.1.2 Arquitetura da Solução

A solução foi desenvolvida seguindo uma arquitetura cliente-servidor, representada na Figura 14. A parte do cliente nesta solução é composto pelas aplicações descentralizadas(Dapps) desenvolvidas, onde um módulo corresponde ao sistema utilizado pelas autoridades de trânsito e o outro módulo sendo o sistema de acesso dos motoristas. Já o servidor da solução corresponde a rede ethereum onde o `deploy` do contrato inteligente foi feito e as transações são validadas pelos nós mineradores da rede, representado na Figura 22.

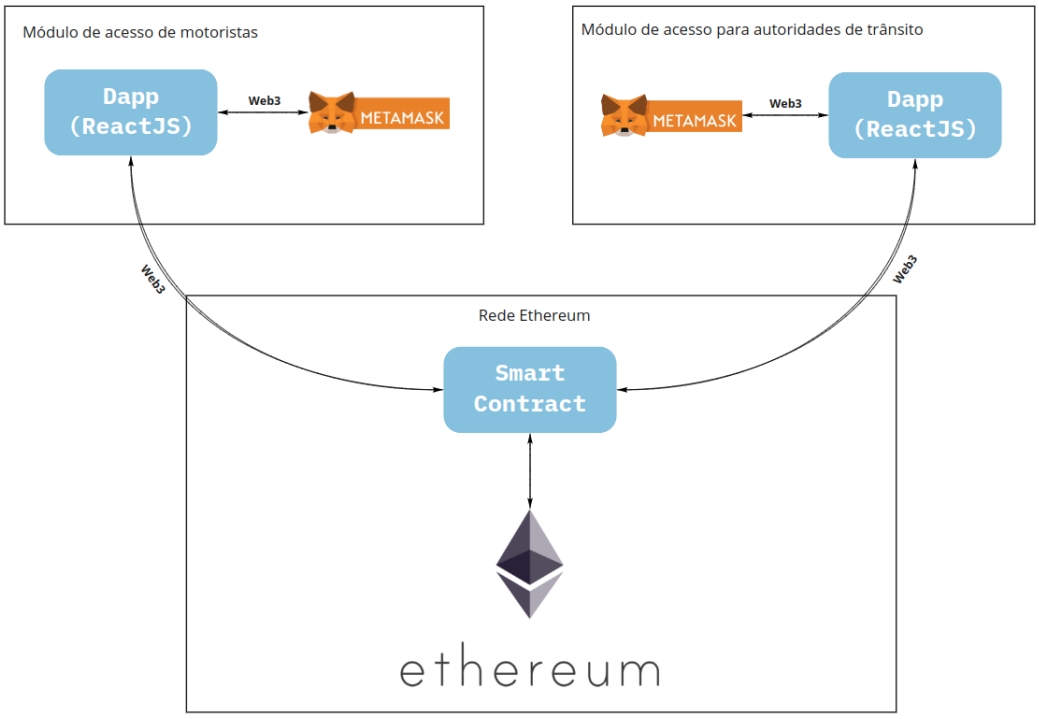


Figura 14 – Representação da interação entre os módulos da solução - Imagem própria.

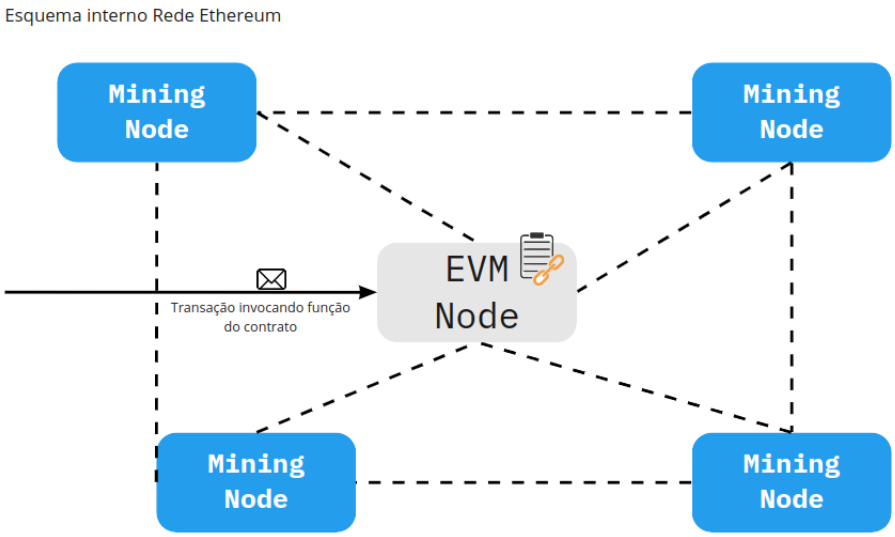


Figura 15 – Representação da rede onde o contrato está na Ethereum Virtual Machine - Imagem própria.

A rede ethereum atualmente utilizada para validar o protótipo e solução desenvolvida é a rede de testes Ethereum Ropsten. A rede de testes Ropsten é executada utilizando os mesmos protocolos da rede principal Ethereum, porém sem custos financeiros atrelados a seu funcionamento podendo assim que testes sejam realizados sem a aquisição de Ethers. A rede provê por meio dos Faucets a possibilidade dos usuários obterem Ethers de forma gratuita e poderem assim usufruir de todas as funcionalidades da rede como realizar transações, deploy de smart contracts, envio de ethers e etc.

5.1.3 Módulos do Sistema

A solução do sistema é composta por dois diferentes módulos, um para acesso do usuário motorista e outro para acesso das autoridades de trânsito. Cada módulo desse sistema corresponde a uma aplicação, sendo estas executadas de forma paralela.

As funcionalidades do sistema possuem especificidades de acordo com o perfil de usuário que está utilizando o sistema, o que demanda um nível de autenticação para acesso a estas informações. A proposta de trabalho traz algumas premissas, onde as premissas II e IV indicam que usuário do sistema seja ele um motorista ou autoridade de trânsito possui uma conta e uma única chave de acesso que o representa. Para realizar a autenticação dentro dos módulos do sistema, estas premissas são utilizadas como base e ocorre utilizando-se esta chave única.

5.1.4 Autenticação dos usuários nos módulos sistema

A autenticação nas aplicações não ocorre por meio de login e senha como usualmente em sistemas. Nesta solução a autenticação é feita com o auxílio da extensão metamask. Optou-se por esta abordagem para que o usuário tenha suas informações ainda mais resguardadas sem a necessidade de armazená-las em um local externo, garantindo assim que o princípio confidencialidade da segurança computacional definido por Stallings esteja presente em meio as aplicações.

O fluxo de autenticação ocorre da seguinte maneira em meio as aplicações:

I. Quando o usuário requisita acesso a uma página do sistema, conta ativa no momento na extensão metamask é obtida com auxílio da extensão Web3;

II. É feita uma chamada de função ao smart contract onde é verificado se esta conta existe em meio aos arrays de motorista ou autoridades de trânsito de acordo com o módulo acessado;

III. Se a conta está presente no array solicitado a página é renderizada e o usuário está assim autorizado no sistema. No caso de negativa desta condição o usuário não é autorizado e não consegue acessar as funcionalidades do sistema.

5.1.5 Autorização dentro do smart contract

Na aplicação possuímos um esquema de autenticação de forma a garantir que somente pessoas autorizadas possuam acesso a determinadas funcionalidades, porém como garantir que isso também ocorra dentro do smart contract que é público dentro da rede

Dentro da rede Ethereum caso uma pessoa tenha o endereço do contrato o mesmo poderá visualizar todas as transações realizadas, assim como realizar chamadas de funções declaradas como públicas no código fonte caso tenha o conhecimento de sua declaração.

Como qualquer pessoa pode interagir com este contrato na rede pública existe outra camada de autenticação dentro das funções públicas desenvolvidas no contrato, onde caso a condição determinada não seja atendida a execução da função solicitada é interrompida. Este controle é feito utilizando a função *require()* presente na linguagem solidity, onde é verificado se o endereço(*address*) da conta *sender* está registrado e autorizado a realizar o procedimento solicitado como no caso do registro de infrações de trânsito que só podem ser feitas por uma conta registrada com o perfil de autoridade de trânsito.

5.1.6 Funcionalidades

A proposta de solução elencou na Seção 3.1.2 algumas funcionalidades a serem desenvolvidas de forma a solucionar o problema levantado no desenvolvimento deste trabalho. A solução desenvolvida conseguiu contemplar todas estas funcionalidades propostas e consideradas fundamentais para o funcionamento do sistema RENAINF, demonstrando a possibilidade de adaptação de um sistema existente utilizando-se a tecnologia blockchain.

Porém tornou-se necessário uma adaptação na funcionalidade de Recurso / Cancelamento de infração de trânsito proposta inicialmente. Ao se mapear os requisitos de forma que essa atenda ao RENAINF e o CTB, ocorreu um erro de interpretação onde a proposta trazia inicialmente que a autoridade de trânsito que realizou o registro da infração seria também responsável por julgar este recurso e isto acaba não seguindo totalmente o processo adotado no código CTB. O Artigo 287 do CTB diz que uma junta de trânsito (Jari) será responsável pelo julgamento do recurso proposto e não a autoridade que registrou a infração, então foi necessário esta adaptação nos requisitos iniciais.

Para o funcionamento desta funcionalidade na solução atual a Jari é considerada uma autoridade de trânsito, que só possui acesso a essa funcionalidade, e é única em todo o sistema. Atualmente, todos os recursos realizados pelos motoristas são direcionados a essa única Jari registrada previamente na criação do contrato, atendendo assim o que está registrado no CTB e corrigindo o requisito errôneo mapeado anteriormente.

5.1.6.1 Aplicação desenvolvida

Para exemplificar do funcionamento dos módulos da aplicação cliente a seguir será descrito o processo para o registro de uma infração de trânsito realizado por uma autoridade de trânsito

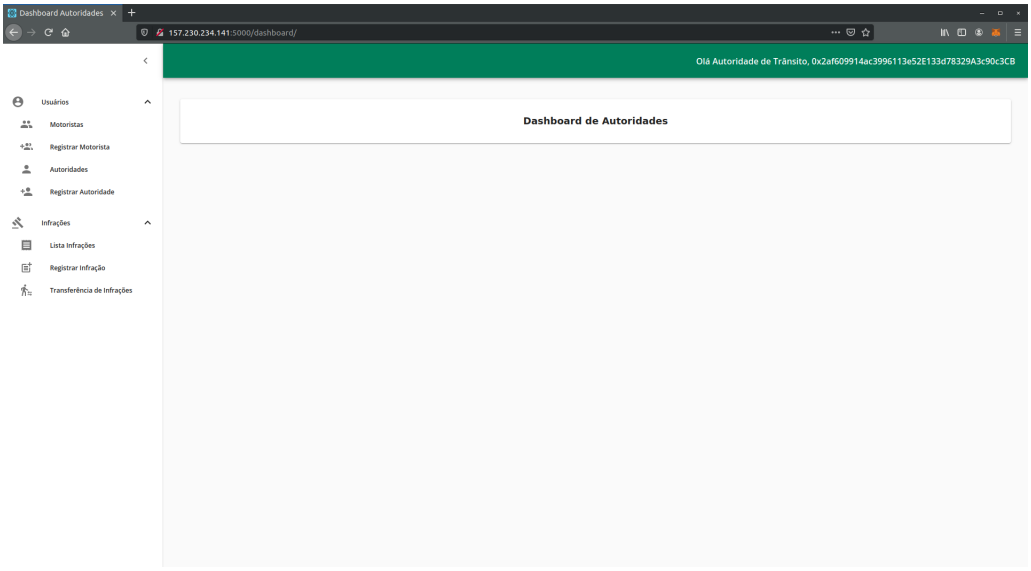


Figura 16 – Autoridade de Trânsito acessa o módulo exclusivo para autoridades de trânsito com a sua carteira autenticada

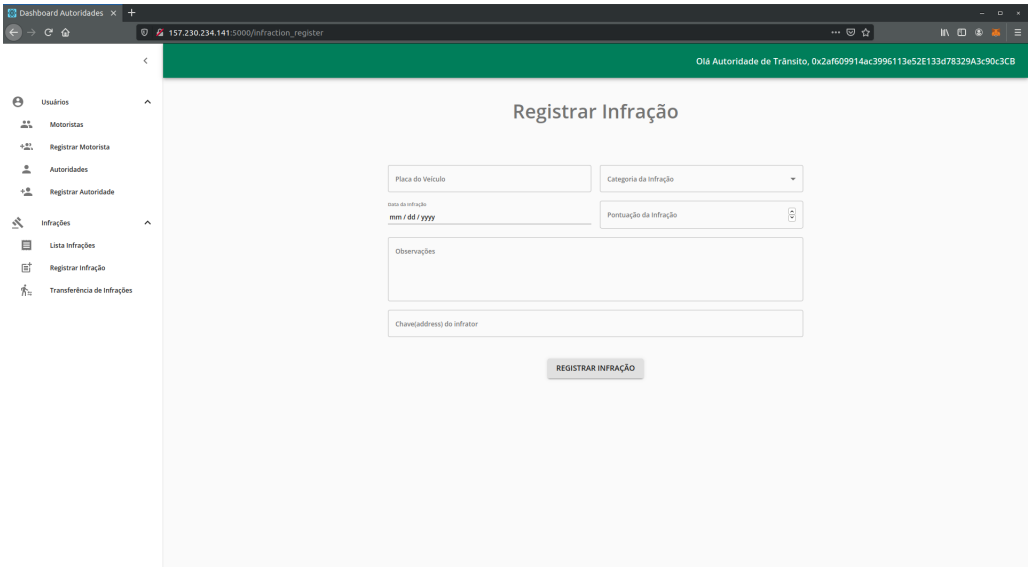


Figura 17 – A página contendo o formulário para registro é acessada utilizando o menu lateral

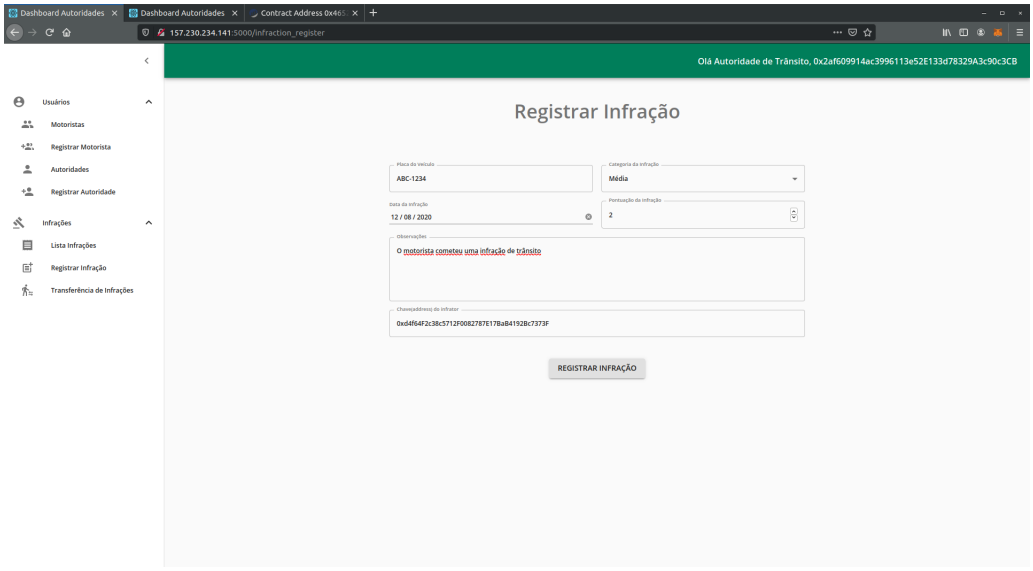


Figura 18 – As informações são preenchidas e o botão Registrar Infração é clicado.

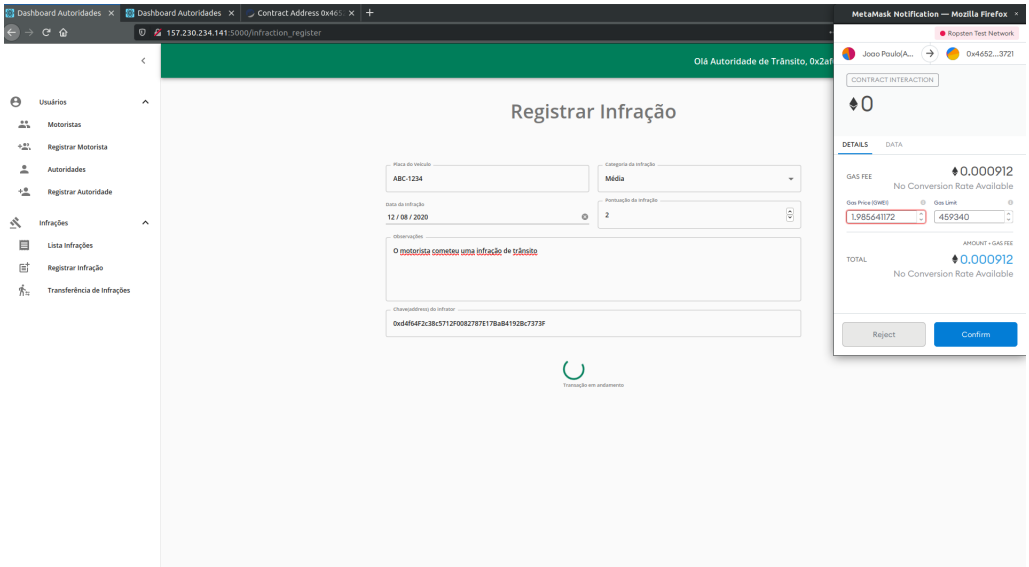


Figura 19 – A transação é assinada utilizando a carteira Metamask

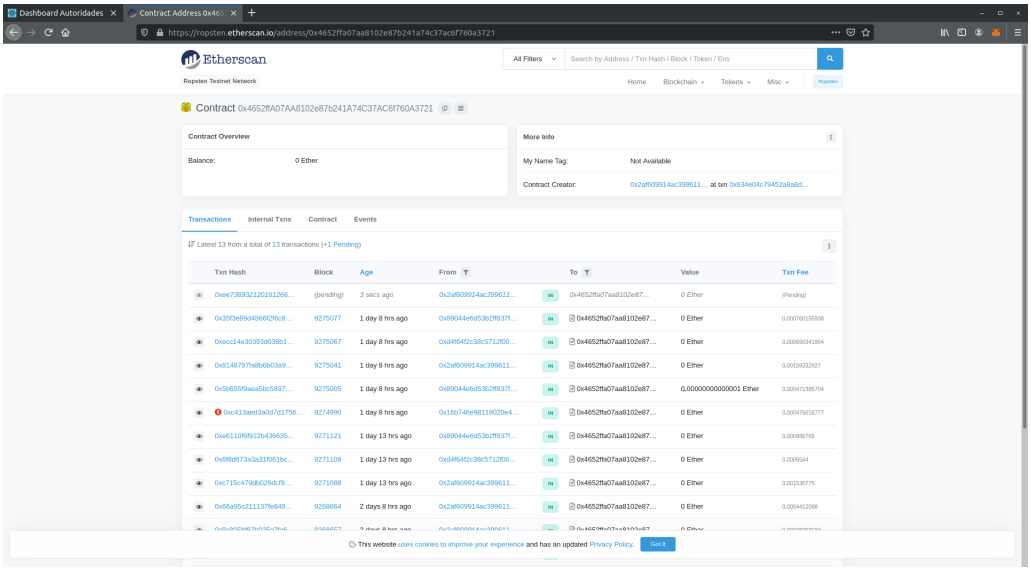


Figura 20 – Uma nova transação é criada para ser validada na rede blockchain.

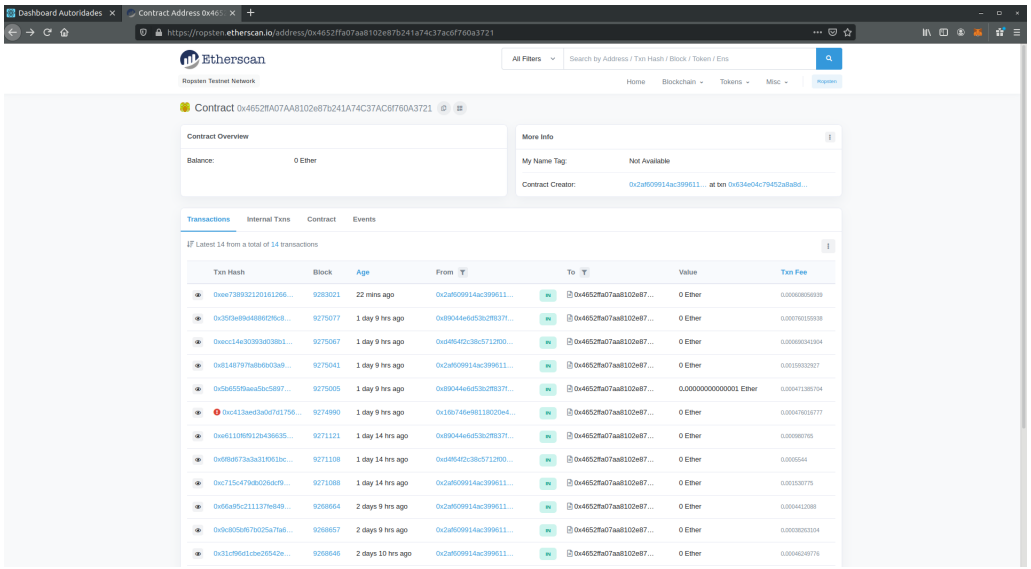


Figura 21 – Transação confirmada com sucesso e inserida na blockchain

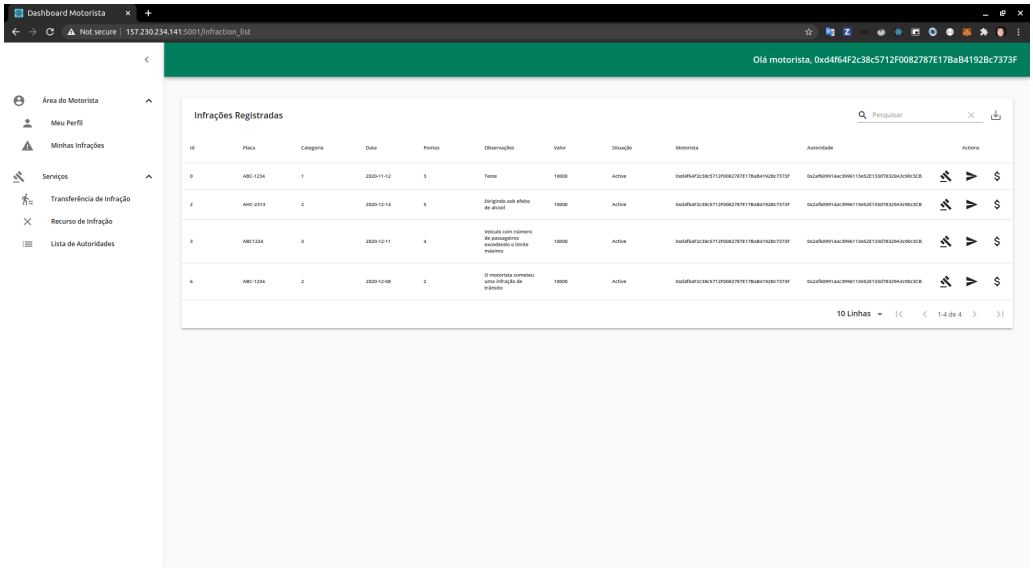


Figura 22 – Infração registrada e listada para o motorista infrator no módulo de motoristas

Para visualizar o funcionamento das outras funcionalidades presentes no sistema assim como especificações de seu funcionamento acesse o vídeo disponível em <https://www.youtube.com/watch?v=Q1qknimuyW8>, ou execute uma instância local da aplicação utilizando o código fonte disponibilizado na Seção 5.3.

5.2 Considerações acerca da solução

O módulo cliente desenvolvido é um protótipo que busca validar o desenvolvimento de uma solução ao problema utilizando aplicações descentralizadas dentro da rede Ethereum. Naturalmente, alguns cuidados necessários a aplicações em produção não foram implementados neste momento, porém que não devem ser desconsiderados na utilização do mesmo em outros contextos.

A solução atual não contempla:

- Requisitos de acessibilidade, não sendo totalmente acessível a navegação por meio de teclado ou leitores de tela;
- As aplicações descentralizadas no momento são executadas sob um protocolo HTTP dentro do servidor, sendo necessária em caso de utilização fora do contexto de validação a utilização do protocolo HTTPS;
- Atualmente o contrato está disponível para utilização em uma Rede Pública de testes e a configuração das aplicações foi feita para este caso. Se desejar utilizar a aplicação cliente em outra rede diferente da Ropsten

Testnet é necessário que se altere essa configuração dentro do arquivo Infraction.json dentro do código fonte de cada aplicação;

5.2.1 Segurança das bibliotecas e ferramentas utilizadas

Para o desenvolvimento da solução foram utilizadas algumas ferramentas e bibliotecas de terceiros, elencadas na Seção 5.1.1, e algumas considerações a cerca da segurança das mesmas precisam ser feitas. A aplicação desenvolvida traz uma segurança e confiabilidade nos dados utilizando-se da tecnologia blockchain, porém também se torna necessário que os módulos da aplicação cliente também sigam critérios de segurança computacional para garantir que a solução como um todo seja segura para uso.

A segurança da aplicação cliente está diretamente ligada as ferramentas e bibliotecas utilizadas para sua construção, assim como os aspectos de implementação. A seguir serão feitas algumas considerações a cerca da segurança destes elementos e a sua utilização na aplicação.

5.2.1.1 ReactJS

O ReactJS é uma biblioteca para criação de interfaces de usuário, sendo uma das bibliotecas mais utilizadas ao redor do mundo para a criação de aplicações web. Seu desenvolvimento é feito de forma open-source, tendo uma grande comunidade ativa evoluindo sua implementação e corrigindo vulnerabilidades quando encontradas trazendo confiabilidade por parte de quem a utiliza, porém é necessário que o desenvolvedor tome alguns cuidados durante a implementação para garantir que as informações estão seguras de possíveis ameaças. O UPPLABS elenca alguns dos riscos mais comuns para aplicações web e aplicações que utilizam o ReactJS, e entre eles temos: *Cross-site Scripting*, *SQL Injection*, estado inicial controlado pelo atacante de renderização do lado do servidor, execução arbitrária de código, etc.

Além de alguns dos problemas mais comuns de segurança entre as aplicações o UPPLABS também traz algumas boas práticas para a segurança de aplicações ReactJS, e entre as boas práticas temos:

- Como defesa contra vulnerabilidades de XSS, remova ou desative qualquer marcação que possa conter instruções para executar o código. Para HTML, isso inclui elementos como <script>, <object>, <embed> e <link>.
- Implemente Idle Timeout na aplicação React.
- Use bibliotecas de snippet como ES7 React, Redux, JS Snippets, etc. Eles trarão segurança adicional e manterão seu código livre de bugs.

- Explore falhas de injeção de script nos aplicativos.
- O código deve se comportar conforme o esperado e deve ser facilmente testável. É recomendado nomear seus arquivos de teste como os arquivos de origem com um sufixo .test.
- Limpe todo o aplicativo React durante e após o processo de desenvolvimento para capturar todos os ataques DDoS de vários tipos.
- Teste a funcionalidade de cada componente da aplicação e teste seu aplicativo completo assim que ele é renderizado no navegador.

Como pode-se notar a segurança da aplicação cliente que utiliza ReactJS para sua construção é obtida a partir de aspectos de segurança computacional implementados dentro da biblioteca assim como boas práticas por parte do desenvolvedor durante a construção da mesma. A solução implementada neste trabalho leva em conta somente alguns destes aspectos em sua implementação pois se trata de um protótipo, o que torna necessário que ao se utilizar este trabalho em um contexto de produção os aspectos citados anteriormente sejam revisados e implementados caso não estejam presentes.

5.2.1.2 Web3

O Web3, como dito anteriormente, é uma coleção de bibliotecas que permitem que você interaja com um nó Ethereum local ou remoto usando HTTP, IPC ou WebSocket. A sua construção também é feita de forma open-source, assim como o ReactJS, tendo um grande comunidade verificando e agindo para corrigir vulnerabilidades encontradas.

Na versão mais atual desta biblioteca até o momento (1.3.1) existe uma vulnerabilidade conhecida que foi reportada pelo SNYK, uma plataforma de desenvolvedores focada em construir aplicações com segurança, sendo descrita como:

> "As versões afetadas deste pacote são vulneráveis ao armazenamento inseguro de credenciais. A implementação atual do web3.js pode resultar na descriptografia da carteira em certas circunstâncias. Quando uma carteira é salva e criptografada no armazenamento local, uma chave privada é necessária para carregar a carteira. No entanto, essa chave privada está disponível via *LocalStorage* e pode ser lida em texto simples em uma página da web depois que uma carteira é carregada.
>
> Essa implementação pode ser abusada por um invasor por meio de ataques do lado do cliente, como *Cross-site Scripting* (XSS), e pode resultar no roubo da chave privada da carteira de um usuário."

Para que essa vulnerabilidade presente nesta versão da biblioteca utilizada no projeto não seja explorada por um atacante é necessário que mecanismos de segurança

sejam implementados na aplicação cliente que a utiliza. Evitando assim que os ataques no cliente possam explorá-la até que uma correção seja publicada em uma versão subsequente da biblioteca.

5.2.1.3 Metamask

O Metamask tem a confiança de mais de um milhões de usuários por todo mundo, sendo essa uma das mais carteiras Ethereum mais populares assim como é considerada uma das melhores disponíveis. Porém o que torna essa carteira segura além da confiabilidade por parte da comunidade que a utiliza?

PELED cita que a carteira MetaMask é uma carteira sem custódia, interagindo diretamente com o blockchain Ethereum. Ele define que uma carteira sem custódia é um tipo de carteira descentralizada, em que o cliente possui suas chaves privadas, onde ao configurar o MetaMask, o usuário obtém um arquivo com as chaves privadas e precisa escrever uma frase-semente com a qual poderá restaurar seus fundos, onde ter chaves privadas significa que você tem controle total sobre os fundos.

Ele também cita que em seu artigo que para maior segurança, MetaMask suporta carteiras de hardware (como Ledger e Trezor): carteiras de hardware podem proteger seus fundos mesmo no caso de uma invasão do computador, tornando a MetaMask uma das opções mais seguras disponíveis.

Porém AARON cita em seu artigo que um risco ao se utilizar carteiras como a Metamask é o ataque de phishing, um tipo de ataque onde o atacante busca roubar informações pessoais como senhas. Ele descreve que este ataque poderia ocorrer da seguinte forma:

> "Gene está trabalhando em seu laptop com várias guias abertas em seu navegador. Ele desbloqueia sua carteira MetaMask para fazer uma transação. Um invasor usa as guias abertas para ver se Gene está usando MetaMask.
>
> O invasor envia a Gene uma mensagem pop-up dizendo que sua transação falhou. Isso acontece às vezes, então Gene não está preocupado. Ele insere sua senha para refazer a transação. O invasor agora tem acesso à carteira de Gene."

Logo em seguida AARON também traz mais informações sobre o assunto relatando que este tipo de ataque é bastante comum em carteiras online e que a equipe do Metamask está trabalhando em patches para este tipo de problema. Dizendo também que ao utilizar carteiras virtuais os usuários são responsáveis por sua própria segurança, trazendo a recomendação de que se use apenas uma guia por vez para fazer transações e mantenha a carteira bloqueada quando não a estiver usando.

5.2.1.4 Docker e Docker-compose

Os contêineres são utilizados para facilitar o gerenciamento de dependências da aplicação assim como orquestrar facilmente os ambientes de desenvolvimento e produção das mesmas. Neste projeto essas ferramentas são utilizadas para facilitar a utilização dos dois ambientes, facilitando a geração de build dos módulos da aplicação cliente e consequentemente o deploy das mesmas no servidor em que estão hospedadas.

KARMAKAR traz em seu artigo algumas vulnerabilidades comuns a contêineres que devem ser observadas e mitigadas pelos desenvolvedores. A seguir vamos elencar algumas:

- Criptojacking: Criptojacking é um tipo de ataque em que um script malicioso é usado para roubar os recursos computacionais de um dispositivo para minerar criptomoedas. Os contêineres podem ser suscetíveis a criptojacking se contiverem imagens maliciosas que dão aos invasores acesso root a todo o contêiner. Eles também são vulneráveis se os endpoints da API do docker container estiverem publicamente acessíveis na Internet sem senhas ou firewalls de segurança.
- Imagens maliciosas de código aberto: É uma vulnerabilidade que torna possível sobrescrever o binário runc do host dá aos atacantes a margem de manobra para executar comandos com acesso root. Os mecanismos Docker anteriores à v18.09.2 tornam os contêineres com imagens controladas pelo invasor suscetíveis à vulnerabilidade CVE-2019-5736. Os utilizadores são aconselhados, tanto quanto possível, a fazer uso de imagens oficiais do Docker fornecidas.
- Dockerfiles estáticos: Um dos princípios dos contêineres é que uma imagem é imutável. Isso significa que, quando uma imagem é construída, seu conteúdo é imutável. Isso por si só dá origem a vulnerabilidades que resultam de pacotes/bibliotecas/imagens desatualizados contidos em uma imagem. Portanto, é uma boa ideia incorporar scanners de vulnerabilidade em processos de CI / CD para identificar imagens de contêiner vulneráveis. Como as imagens são imutáveis, lançar um contêiner recém-criado com dependências atualizadas ajudará a conter as vulnerabilidades de segurança, pois a correção do contêiner é desencorajada.

Neste projeto foram utilizadas as bibliotecas em suas versões mais recentes de forma a mitigar problemas de segurança, imagens oficiais disponibilizadas pela equipe Docker, assim como somente as portas necessárias de cada aplicação são liberadas para acesso externo na configuração dos contêineres. Recomenda-se em uma utilização poste-

rior da aplicação que essas dependências sejam verificadas e atualizadas caso necessário, criando-se assim uma nova imagem com essas atualizações.

5.2.1.5 Solidity

Em toda a linguagem de programação aspectos de segurança devem ser levados em conta e adotados durante a utilização. No Solidity, isso é ainda mais importante porque você pode usar contratos inteligentes para lidar com tokens ou, possivelmente, coisas ainda mais valiosas onde as ações podem ser irreversíveis. Além disso, toda execução de um contrato inteligente acontece em público e, além disso, o código-fonte costuma estar disponível.

A documentação oficial da linguagem traz uma lista de recomendações para os desenvolvedores assim como uma explanação a cerca de mecanismos existentes na linguagem que podem ser utilizados nos contratos de forma a garantir a segurança em sua utilização. Estas recomendações podem ser acessadas em <https://docs.soliditylang.org/en/v0.6.12/security-considerations.html>.

5.3 Acesso a implementação da solução

A solução foi desenvolvida em cima de ferramentas gratuitas e open source, assim como o código fonte utilizado para criação dos Dapps e o código do contrato estão regidos sob a licença GPL3 sendo assim gratuita a sua utilização.

Os arquivos fonte desta solução podem ser encontrados no repositório: <https://github.com/joaopaulonsoares/aplicacaotccblockchain>.

Para dúvidas a cerca da implementação, entre em contato no email paulonsoares18@gmail.com ou criando uma Issue no repositório do Github citado anteriormente.

6 Conclusão

O objetivo deste trabalho é validar a possibilidade de desenvolver uma aplicação descentralizada, por meio da tecnologia blockchain, que garantisse a integridade dos dados relacionados ao registro de infrações de trânsito e outros casos previstos no Código de Trânsito Brasileiro assim como demonstrar a viabilidade da utilização de uma aplicação descentralizada em um contexto hoje totalmente centralizado.

A solução desenvolvida foi implementada utilizando-se de ferramentas e tecnologias atuais, além de seguir padrões de código e arquiteturais amplamente conhecidos pela comunidade de software, podendo ser assim ser facilmente escalada e aprimorada ao longo do tempo para a utilização em outros contextos.

A utilização de contratos inteligentes para o registro de infrações de trânsito traz fatores positivos a este contexto, como por exemplo uma maior auditabilidade das informações registradas assim como uma integridade dos dados dentro do sistema, trazendo uma maior confiabilidade da população em frente aos órgãos responsáveis pelo trânsito no Brasil. Porém a utilização do contrato utilizando esta tecnologia neste contexto também pode esbarrar em um problema caso ocorra a modificação da constituição que hoje rege esse sistema pois os contratos são imutáveis e não podem ser alterados após a sua criação, fazendo assim com que uma mudança mesmo que simples possa levar a necessidade da criação de um novo contrato e assim todas as transações e dados armazenados no contrato anterior precisem ser migrados ou uma camada intermediária seja criada para adaptar esse novo contexto.

Outro fator de atenção na utilização de uma rede descentralizada neste contexto é o tempo para a confirmação de uma transação solicitada, como por exemplo o registro de uma infração. Este tempo é determinado pelo tamanho e tipo da rede blockchain utilizada, sendo assim importante analisar o contexto e avaliar o que melhor se aplica podendo este problema ser mitigado pela utilização de uma rede robusta ou por meio de uma melhor experiência de usuário dentro da aplicação cliente caso seja necessário um intervalo considerado grande até a confirmação de uma transação.

A solução desenvolvida validou o objetivo proposto demonstrando que apesar de possíveis problemas, que são inerentes a todos os tipos de aplicações, é possível a utilização da tecnologia blockchain em um contexto de registro de infrações de trânsito que na data de publicação deste trabalho é considerado burocrático, ineficiente e centralizado sem que hajam prejuízos significativos a seus utilizadores.

A solução pode ser abstraída para outros contextos da esfera pública de administração, onde a criação de uma rede blockchain governamental possibilitaria que diversos

serviços fossem disponibilizados de forma descentralizada facilitando o acesso e a utilização do mesmo por meio da população. E por meio de incentivos governamentais essa rede poderia ser expandida para diversos nós trazendo mais integridade, confiabilidade e auditabilidade aos dados de diversos órgãos.

Referências

AARON. *MetaMask Wallet Review*. 2020. Disponível em: <https://www.bitdegree.org/crypto/metamask-wallet-review#metamasknbspsecuritynbsp>. Acesso em: 17/12/2020. Citado na página 64.

ALLEN, J. J. B. P. R. *Blockchain, A Practical Guide to Developing Business, Law, and Technology Solutions*. [S.l.]: Mc Graw Hill Education, 2018. ISBN 9781260115871. Citado 8 vezes nas páginas 11, 31, 32, 33, 34, 38, 39 e 40.

ANTONOPOULOS, A. M. *Mastering Bitcoin,Programming the Open Blockchain - 2 edição*. O'Reilly, 2017. Disponível em: <http://oreilly.com/catalog/errata.csp?isbn=9781491954386>. Citado 4 vezes nas páginas 29, 32, 34 e 35.

BASHI, I. *Mastering Blockchain*. [S.l.]: Packt Publishing, 2017. Citado 8 vezes nas páginas 11, 24, 25, 30, 33, 34, 35 e 36.

BREWER, E. Towards robust distributed systems. In: . [S.l.: s.n.], 2000. p. 7. Citado na página 30.

BUTERIN, V. Ethereum white paper,a next generation smart contract decentralized application platform. 2013. Disponível em: <http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf>. Citado 2 vezes nas páginas 38 e 39.

FAZENDA E PLANEJAMENTO DE SãO PAULO. *Mais informações Renainf*. 2018. Disponível em: <https://portal.fazenda.sp.gov.br/servicos/ipva/Paginas/mi-renainf.aspx>. Acesso em: 05 jun. 2019. Citado 2 vezes nas páginas 18 e 44.

HERLIHY, M. Blockchains from a distributed computing perspective. *Communications of the ACM*, v. 62, n. 2, p. 78 – 85, 2019. ISSN 00010782. Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=134383087&lang=pt-br&site=eds-live>. Citado na página 32.

HOFFMANN MARIA HELENA; CARBONELLI, E. M. L. Álcool e segurança no trânsito (ii): a infração e sua prevenção. *Psicol. cienc. prof. [online]*, v. 16, n. 2, p. 25–30, 1996. ISSN 1414-9893. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1414-98931996000200006&lng=en&nrm=is>. Citado na página 17.

JOKIć, S. et al. Comparative analysis of cryptocurrency wallets vs traditional wallets. In: . [S.l.: s.n.], 2019. v. 65. Citado na página 41.

KARMAKAR, D. *How to find and fix Docker container vulnerabilities in 2020*. 2020. Disponível em: <https://www.freecodecamp.org/news/how-to-find-and-fix-docker-container-vulnerabilities-in-2020/>. Acesso em: 17/12/2020. Citado na página 65.

LAMPORT LESLIE ;SHOSTAK, R. . P. M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems,*, v. 4, n. 3, p. 382 – 401, 1982. Disponível em: <http://www-inst.eecs.berkeley.edu/~cs162/fa12/hand-outs/Original_Byzantine.pdf>. Citado na página 31.

LAURENCE, T. *Blockchain For Dummies ® , 2nd Edition*. [S.l.]: John Wiley Sons, Inc, 2019. ISBN 978-1-119-55501-8. Citado 2 vezes nas páginas 11 e 36.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Citado na página 32.

NIST. *An Introduction to Information Security*. [S.l.], 1995. Citado na página 21.

PANDA, B. S. G. D. . P. S. *Beginning Blockchain, A Beginner's Guide to Building Blockchain Solutions*. Apress, 2018. ISBN 781484234440. Disponível em: <https://doi.org/10.1007/978-1-4842-3444-0>. Citado 11 vezes nas páginas 11, 18, 21, 22, 23, 25, 27, 28, 29, 36 e 38.

PELED, E. *MetaMask Security Overview and Connecting to Tetra Wallet*. 2020. Disponível em: <https://www.orbs.com/metamask-security-overview-and-connecting-to-tetra-wallet/>. Acesso em: 17/12/2020. Citado na página 64.

PRESIDêNCIA DA REPúBLICA. *Código de Trânsito Brasileiro, Lei 9.503 de 23 de Setembro de 1997*. 1997. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/l9503.htm>. Acesso em: 05 jun. 2019. Citado 2 vezes nas páginas 41 e 43.

ROSIC, A. *Cryptocurrency Wallet Guide: A Step-By-Step Tutorial*. 2018. Disponível em: <https://blockgeeks.com/guides/cryptocurrency-wallet-guide/>. Acesso em: 17/12/2020. Citado na página 40.

SHARMA, A.; S.K.MITTAL. Attacks on cryptographic hash functions and advances. v. 5, p. 89–96, 11 2018. Citado na página 27.

SNYK. *Web3 vulnerabilities, Ethereum JavaScript API*. 2020. Disponível em: <https://snyk.io/vuln/npm:web3>. Citado na página 63.

STALLINGS, W. *Cryptography and Network Security Principles and Practice - Fifth Edition*. [S.l.]: Prentice Hall, 2011. ISBN 9780136097044. Citado 6 vezes nas páginas 11, 21, 22, 23, 24 e 26.

STAPLES, X. X. . I. W. . M. *Architecture for Blockchain Applications*. [S.l.]: Springer, 2019. ISBN 978-3-030-03034-6. Citado na página 37.

SZABO, N. Smart contracts: Building blocks for digital markets. 1996. Disponível em: <www.alamut.com/subj/economics/nick_szabo/smartContracts.html>. Citado na página 38.

TANENBAUM ANDREW S.; STEEN, M. v. *Sistemas Distribuídos: princípios e paradigmas - 3ª edição.* Pearson, n.d. ISBN 9788576051428. Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&db=cat07297a&AN=unb.9788576051428&lang=pt-br&site=eds-live>. Citado na página 30.

UPPLABS. *REACT.JS SECURITY BEST PRACTICE*. 2020. Disponível em: <https://upplabs.com/blog/react-js-security-best-practices/>. Citado na página 62.

WANG YIQUN LISA YIN, F. Y. X. Finding collisions in the full sha-1. 2005. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Citado na página 27.

Apêndices

APÊNDICE A – Structs do contrato inteligente

```
struct TraficTicket{
    uint id;
    string vehiclePlate;
    uint infractionCategory;
    string dateInfraction;
    uint infractionPoints;
    string observations;
    uint valueToPay;
    string statusOfInfraction;
    bool payed;
    address infractorDriverAddress;
    address payable authorityResponsableAdress;
}

struct Drivers{
    uint id;
    string name;
    uint numberOfPoints;
    address driverAddress;
    bool exist;
}

struct Authorities{
    uint id;
    string name;
    string sigla;
    address authoritieAddress;
    bool exist;
    bool isTransitBoard;
}

struct TransferTicket{
    uint id;
```

```
        uint ticketId;
        uint status; // 0 -> Pending, 1 -> Accepted, 2 -> Rejected
        address currentOwnerAddress;
        address requestedInfractorAddress;
    }

    struct CancelTickeRequest{
        uint id;
        uint ticketId;
        uint status; // 0 -> Pending, 1 -> Accepted, 2 -> Rejected
        string explanation;
        address driverAddress;
        address authorityResponsableAdress;
        address transitBoardResponsible;
    }
```