UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Franciel Augusto Haas Krein

PROTÓTIPO E ANÁLISE DE SISTEMA DE VOTAÇÃO ELETRÔNICO BASEADO EM BLOCKCHAIN

Santa Maria, RS
2018

Franciel Augusto Haas Krein

PROTÓTIPO E ANÁLISE DE SISTEMA DE VOTAÇÃO ELETRÔNICO BASEADO EM BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Sistemas de Informação.

ORIENTADOR: Prof. Joaquim Vinicius Carvalho Assunção

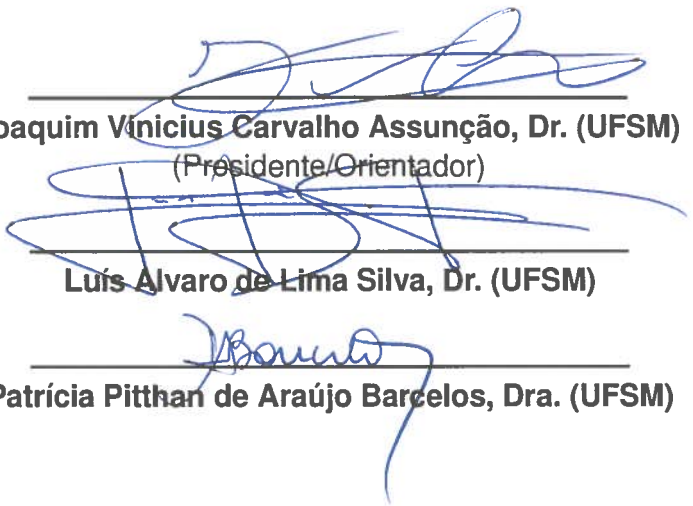
Santa Maria, RS
2018

Franciel Augusto Haas Krein

PROTÓTIPO E ANÁLISE DE SISTEMA DE VOTAÇÃO ELETRÔNICO BASEADO EM BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Sistemas de Informação.

Aprovado em 4 de dezembro de 2018:



Joaquim Vinicius Carvalho Assunção, Dr. (UFSM)
(Presidente/Orientador)

Luís Álvaro de Lima Silva, Dr. (UFSM)

Patrícia Pitthan de Araújo Barcelos, Dra. (UFSM)

Santa Maria, RS
2018

RESUMO

PROTÓTIPO E ANÁLISE DE SISTEMA DE VOTAÇÃO ELETRÔNICO BASEADO EM BLOCKCHAIN

AUTOR: Franciel Augusto Haas Krein
ORIENTADOR: Joaquim Vinicius Carvalho Assunção

Sendo o voto a base de um governo democrático, é imperativo utilizar todos os meios disponíveis para garantir sua integridade. O armazenamento e gerenciamento dos votos de maneira centralizada remove a transparência dos processos de votação, incumbindo a sua auditoria exclusivamente às entidades gestoras destes processos, além de introduzir pontos centrais de ataque, que, caso sejam explorados com sucesso por agentes mal-intencionados, podem comprometer todo o resultado da eleição. Este projeto propõe o desenvolvimento de um protótipo para um sistema de votação baseado em urnas eletrônicas, onde os votos são armazenados de maneira descentralizada, permitindo que qualquer eleitor audite o processo eleitoral. Para assegurar a integridade das informações, os votos são armazenados em uma blockchain pública, sendo a proveniência destes votos garantida por algoritmos de assinatura digital criptográficos. Realiza-se uma análise das vantagens e limitações deste protótipo em relação às abordagens tradicionais.

Palavras-chave: Votação. Eleição. Blockchain. Segurança da Informação.

ABSTRACT

PROTOTYPE AND ANALYSIS OF AN ELECTRONIC VOTING SYSTEM BASED ON BLOCKCHAIN

AUTHOR: Franciel Augusto Haas Krein
ADVISOR: Joaquim Vinicius Carvalho Assunção

Since votes are the foundation of any democratic government, it is imperative to utilize all available tools to guarantee their integrity. The centralized storage and management of votes removes the transparency of the voting processes, entrusting their auditing exclusively to the entities managing these processes, besides introducing central points of attack, which, if successfully exploited by malicious agents, can compromise the entire election result. This project proposes the development of a prototype for a voting system based on electronic voting machines, where votes are stored in a decentralized manner, allowing any voter to audit the election process. To guarantee the information integrity, votes are stored in a public blockchain, being the the provenance of these votes ensured by cryptographic digital signature algorithms. An analysis is performed on the advantages and limitations of this prototype in contrast to the traditional approaches.

Keywords: Voting. Election. Blockchain. Information Security.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
DoS	Denial of Service
$DDoS$	Distributed Denial of Service
$ECDSA$	Elliptic Curve Digital Signature Algorithm
$HTTP$	HyperText Transfer Protocol
IP	Internet Protocol
NAT	Network Address Translation
PoW	Proof of Work
RSA	Rivest–Shamir–Adleman
UML	Unified Modeling Language
URL	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Em decorrência da grande quantidade de benefícios que podem estar associados ao resultado de uma eleição, não é incomum encontrarem-se exemplos de processos eleitorais em que foram empregadas técnicas para a sua manipulação[1]. Estas podem abranger desde a coerção dos eleitores até a adulteração dos resultados pela entidade gestora do processo[2]. Estes processos são facilitados quando são introduzidos dispositivos eletrônicos para o registro ou contagem dos votos, visto que um único atacante pode, facilmente, comprometer todo o resultado da eleição. A utilização de sistemas totalmente digitais dificulta, ainda, o processo de auditoria da votação, vista a complexidade de determinar se um registro é autêntico ou sofreu alteração. Este processo é imprescindível para a determinação da validade dos resultados (LOWRY; VORA, 2009).

Aliadas a estes elementos, tem-se as contínuas descobertas de vulnerabilidades nas urnas eletrônicas utilizadas nas eleições brasileiras, conforme apontado por Aranha et al. (2018), onde os pesquisadores conseguiram desde a recuperação de chaves criptográficas até a injeção de códigos arbitrários na urna, limitando a confiabilidade do sistema.

Nos modelos eleitorais tradicionais, o gerenciamento da votação por uma entidade centralizada remove, ainda, a transparência do processo. O eleitor não possui qualquer garantia da validade dos resultados apresentados, restando-lhe apenas confiar nas informações providas. A gestão desse processo de forma centralizada acarreta também na existência de pontos centrais de coleta e contagem de votos, que podem ser explorados por um agente comprometido, seja este gestor do processo eleitoral ou um atacante externo. Logo, o comprometimento deste ponto pode implicar no comprometimento de toda a eleição.

Considerando o papel do voto em um governo democrático, é imperativo que os sistemas que permeiam o processo eleitoral sejam capazes de assegurar a integridade das informações, garantindo ao eleitor, de forma transparente, a ausência de fraude em todas as etapas do processo.

Com o advento da *blockchain* em 2008 (NAKAMOTO, 2008) e a sua ampla utilização em criptomoedas (dentre as quais destaca-se a *Bitcoin*), notou-se o surgimento de sistemas de votação baseados na tecnologia, que descentralizam o processo de votação, transferindo a manutenção da informação aos próprios *peers* conectados à rede. Dentre esses sistemas, citam-se o *Follow my Vote* e o *VoteWatcher* (OSGOOD, 2016).

Dentro deste contexto, este trabalho busca analisar as abordagens adotadas por estes sistemas, propondo o desenvolvimento de um protótipo para um sistema de votação baseado em urnas eletrônicas, onde os votos são armazenados em uma *blockchain* pú-

[1] Wikipedia: https://en.wikipedia.org/wiki/List_of_controversial_elections

[2] NIST: https://web.archive.org/web/20061021021000/http://vote.nist.gov/threats/papers.htm

blica. Para determinar que os votos provêm de urnas válidas, são utilizadas assinaturas digitais criptográficas, enquanto que o mecanismo de consenso adotado pelos *peers* da rede para a determinação da ordem de inserção dos blocos consiste em um protocolo de Prova de Trabalho baseado em *hashes*. Realiza-se uma análise das implicações da utilização deste sistema, buscando determinar suas vantagens e desvantagens em relação aos sistemas de votação tradicionais.

Este documento é dividido em cinco capítulos. O capítulo 2 apresenta a fundamentação teórica do trabalho, definindo as técnicas computacionais utilizadas para o desenvolvimento do protótipo. Ainda neste capítulo, são analisados sistemas similares ao proposto, que empregam *blockchain* para o armazenamento de votos durante eleições. Ao longo do capítulo 3, é descrita a arquitetura do protótipo e a metodologia adotada para seu desenvolvimento. O capítulo 4 analisa as implicações do uso do sistema proposto em eleições. O capítulo 5 discorre sobre os resultados e apresenta as considerações finais acerca do trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo são definidos os conceitos utilizados ao longo deste trabalho. A seção 2.1 define funções *hash* criptográficas e discorre sobre suas propriedades. Em seguida, na seção 2.2, são abordadas assinaturas digitais criptográficas. A seção 2.3 define *blockchains*, sendo provida uma descrição de seu funcionamento. A seção final deste capítulo analisa os sistemas *Follow my Vote* e *VoteWatcher*, que empregam *blockchains* para o armazenamento dos votos durante eleições.

2.1 FUNÇÕES HASH CRIPTOGRÁFICAS

Conceitua-se como uma função *hash* qualquer função determinística que mapeia um dado de tamanho arbitrário para uma *string* de comprimento fixo (onde a *string* retornada é denominada *hash*). Para categorizar uma função *hash* como criptográfica, esta deve ser projetada de modo a ser uma função de sentido único (não possuindo função inversa), onde o cálculo da *hash* a partir do valor de entrada pode ser realizado rapidamente, porém, a única maneira de obter-se o valor de entrada a partir do valor de saída é através da realização de uma busca por força bruta, testando todos os possíveis valores de entrada até encontrar uma correspondência. Idealmente, qualquer mudança, por menor que seja, no valor de entrada, deve alterar completamente a *hash* produzida, de modo que o novo valor aparente não ter qualquer correlação com o valor antigo.

O nível de segurança de uma função *hash* criptográfica é medido levando-se em consideração três propriedades (ROGAWAY; SHRIMPTON, 2004 apud KATZ et al., 1996):

- Resistência à pré-imagem - Para qualquer *hash*, é computacionalmente inviável encontrar um valor de entrada que a produza;
- Resistência à segunda pré-imagem - Para qualquer valor de entrada, é computacionalmente inviável encontrar um segundo valor de entrada que produza a mesma *hash*;
- Resistência à colisão - É computacionalmente inviável encontrar dois valores de entrada que produzam o mesmo valor de saída.

Hashes criptográficas constituem uma das bases da *blockchain*, sendo que os protocolos que a compõem são construídos sob a presunção da validade destas propriedades.

2.2 ASSINATURAS DIGITAIS CRIPTOGRÁFICAS

Assinaturas digitais criptográficas são protocolos matemáticos que permitem a verificação da autenticidade de informações digitais. O processo de assinatura inicia-se com a geração de um par de chaves criptográficas assimétricas (uma chave pública e uma chave privada). O algoritmo que gera a assinatura recebe como entradas a mensagem a ser assinada e a chave privada do seu transmissor. Dada a dependência da assinatura sobre o conteúdo da mensagem, é impossível para um agente utilizar a mesma assinatura para validar uma mensagem diferente. Um segundo algoritmo é utilizado pelos receptores da mensagem para verificar a proveniência da assinatura. Este algoritmo recebe como entrada a mensagem, a assinatura e a chave pública do transmissor, e retorna um valor booleano indicando se a assinatura foi gerada pela chave privada que corresponde à chave pública em sua posse.

A utilização de assinaturas digitais permite que qualquer agente possa verificar, com razoável confiabilidade, que uma mensagem provém de um remetente específico e não foi gerada por um terceiro agente interceptando a comunicação. A assinatura provê, ainda, um mecanismo que impossibilita o transmissor da mensagem de negar sua assinatura, visto que somente alguém de posse de sua chave privada poderia gerá-la.

Para ser utilizado como um mecanismo de assinatura digital, um algoritmo deve garantir que a assinatura gerada pela chave privada possa ser verificada utilizando a chave pública correspondente e que seja computacionalmente inviável a geração de uma assinatura válida para qualquer agente que não tenha posse da chave privada.

O conceito de assinaturas digitais foi primeiramente descrito por Diffie e Hellman (1976), sendo o primeiro algoritmo que as implementava (RSA) publicado em 1978. O algoritmo RSA (e seus derivados) baseia sua confiabilidade na premissa de que a fatoração, principal processo utilizado para encontrar-se a chave privada a partir dos dados públicos no RSA, torna-se um processo computacionalmente inviável conforme o tamanho do valor fatorado aumenta (RIVEST; SHAMIR; ADLEMAN, 1978).

Abordagens mais recentes, como algoritmos de assinatura digital de curva elíptica (ECDSA), baseiam seu nível de segurança na premissa de que a computação de logaritmos discretos é custosa, visto que não existe solução polinomial conhecida (MILLER, 1985). Como os mecanismos para encontrar a chave privada em algoritmos de assinatura digital de curva elíptica são menos eficientes que a técnica de fatoração utilizada em algoritmos RSA, um ECDSA consegue prover o mesmo nível de segurança de algoritmos RSA, utilizando chaves significativamente menores.

Assinaturas digitais criptográficas são comumente utilizadas em transações financeiras, como forma de garantir sua proveniência, provendo, ainda, um mecanismo que impede o executor da transação de negar sua participação nesta, uma vez que para afirmar isso, ele teria que declarar a sua chave privada como comprometida.

2.3 BLOCKCHAINS

A *blockchain* foi introduzida como plataforma para a criptomoeda *Bitcoin*, proposta em 2008 por Satoshi Nakamoto, que compreende um conjunto de protocolos para transações monetárias *peer-to-peer*. Cada transação envolvendo *Bitcoins* é protegida por uma assinatura digital gerada a partir da chave privada do usuário realizando a transferência. Essa assinatura pode ser verificada pelos demais *peers* da rede através da chave pública deste usuário, garantindo a proveniência da transação. Cada *peer* verifica, ainda, se o emissor da transação possui fundos suficientes para completá-la. As transações válidas são inseridas em blocos, que podem então ser anexados ao final da *blockchain* (NAKAMOTO, 2008).

A abordagem distribuída deste sistema, no entanto, implica que, em dado momento, diferentes *peers* terão acesso a diferentes transações, o que possibilita a um usuário mal-intencionado gastar mais de uma vez a mesma moeda, divulgando transações distintas para diferentes *peers* conectados à rede. Quando os *peers* estão em comum acordo sobre a ordem dos blocos, eles são capazes de detectar que a moeda já foi gasta em uma transação presente em um bloco anterior, descartando a transação duplicada. Surge, então, a necessidade de um mecanismo que garanta o consenso entre todos os *peers* participantes, a fim de determinar quais transações serão inseridas no próximo bloco da cadeia. Este é o problema endereçado pela *blockchain*.

O bloco inicial da *blockchain*, conhecido como *genesis* na literatura[1], é, geralmente, inserido estaticamente no momento que um *peer* é instanciado. O propósito deste bloco inicial é garantir que todos os *peers* possuam o primeiro bloco em comum. Para determinar o próximo bloco a ser anexado à *blockchain*, o algoritmo de consenso utilizado pela *Bitcoin* requer que este bloco contenha a solução para um problema matemático de alta demanda computacional. O *peer* que conseguir resolver esse problema determina o próximo bloco da cadeia. No eventual cenário de múltiplos *peers* resolverem o problema simultaneamente, cada um deles irá divulgar uma versão diferente da *blockchain*. Nesse cenário, o *peer* que conseguir resolver o próximo bloco determinará qual das versões deve ser utilizada. Os *peers* conectados à rede sempre aceitam a versão mais longa da *blockchain* como válida, visto que esta possui uma quantidade maior de trabalho computacional aplicada (NAKAMOTO, 2008).

O mecanismo adotado pela *Bitcoin* para o consenso é denominado Prova de Trabalho (PoW). Cada *peer* participante no processo cria um novo bloco composto pelas transações válidas que ele recebeu mas ainda não foram adicionadas à *blockchain*. O objetivo do algoritmo é encontrar um valor *nonce* que, ao passar por uma função *hash* criptográfica, produza uma *hash* que inicie com uma quantidade específica de zeros. Como a *hash* produzida é totalmente imprevisível, a maneira mais eficiente para encontrar este valor é

[1] Bitcoin Wiki: https://en.bitcoin.it/wiki/Genesis_block

testando por força bruta todos os valores possíveis. O primeiro *peer* que conseguir achar um *nonce* que produza uma *hash* aceitável tem seu bloco adicionado pelos demais *peers* da rede. Para evitar que um *peer* simplesmente copie o *nonce* encontrado por outro e o adicione ao seu próprio bloco, a função *hash* recebe como entrada, além deste *nonce*, as transações do bloco, garantindo que a *hash* somente seja válida para aquele bloco em específico. O terceiro parâmetro passado para a função *hash*, é a *hash* do bloco anterior da *blockchain*, construindo uma cadeia onde cada bloco é dependente de todos os blocos anteriores a ele. Qualquer tentativa de alterar um bloco inserido requer que todas as *hashes* posteriores a este sejam recalculadas. A dificuldade do algoritmo aumenta exponencialmente conforme a quantidade de zeros necessária no início da *hash* é incrementada. Como a quantidade de poder computacional dos *peers* conectados à rede é dinâmica, a *Bitcoin* ajusta a sua dificuldade dinamicamente para que um bloco novo seja inserido, em média, a cada 10 minutos (NAKAMOTO, 2008).

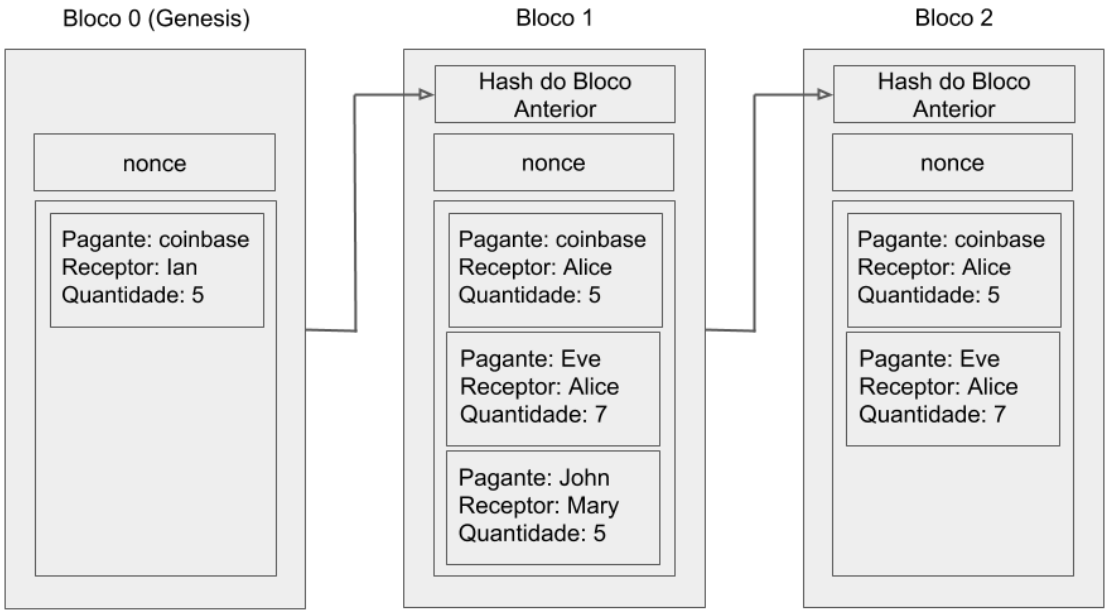
Para um *peer* (ou um conjunto de *peers*) comprometido conseguir modificar a *blockchain* é necessário que este seja capaz de produzir blocos válidos (que apresentam *nonce* satisfatório) a uma velocidade superior à dos demais blocos da rede. Neste cenário, este *peer* passa a possuir a versão mais longa da *blockchain*, a qual é adotada pelos demais *peers* na rede. Para conseguir resolver o PoW antes dos demais *peers* da rede, é necessário que o *peer* comprometido tenha um poder computacional superior ao restante da rede.

A Figura 2.1 apresenta uma versão simplificada da estrutura da *blockchain* utilizada para o armazenamento de transações em criptomoedas. Observa-se que cada bloco é composto por uma lista de transações, um *nonce* e uma *hash* gerada a partir do bloco anterior. Esta estrutura constitui a base para qualquer aplicação baseada em *blockchain*, sendo que, para o armazenamento de dados que não sejam referentes a transações financeiras, basta alterar o tipo de informação armazenada na lista de transações.

Generalizando, define-se como *blockchain* um registro distribuído cuja integridade é preservada por diversos *peers* através de um protocolo criptográfico distribuído, sem uma autoridade central. Todos os *peers* validam a informação a ser anexada à *blockchain*, e um protocolo de consenso garante que os *peers* concordem em uma única ordem em que as entradas são anexadas. Além do PoW, existem diversos protocolos propostos que buscam garantir o consenso entre os *peers* de maneiras diferentes (CACHIN; VUKOLIC, 2017).

A arquitetura da *blockchain* garante a imutabilidade da informação mesmo diante de nós defeituosos ou corrompidos dentro da rede (CACHIN; VUKOLIC, 2017). Essa característica torna sua utilização adequada em uma variedade de aplicações distribuídas onde existe a demanda de que todos os nós estejam de acordo em relação aos recursos que armazenam. Crosby et al. (2016) cita dentre os setores que utilizam *blockchains* o financeiro, musical, de seguros, armazenamento distribuído, entre outros. Em contraste, no entanto, a utilização de sistemas baseados em *blockchain* introduz diversos desafios,

Figura 2.1 – Abstração da blockchain de uma criptomoeda.



Fonte: Produção do próprio autor.

dentre os quais pode-se citar a falta de regulamentação, que possibilita atividades fraudulentas, e a ausência de uma entidade central que ateste e se comprometa pela qualidade do serviço prestado. A tecnologia apresenta ainda limitações de escalabilidade. Os *peers* que mantém a *blockchain* são (geralmente) obrigados a realizar o download de todas as informações que a rede armazena e validá-las, agregando custos de armazenamento e consumo de banda de rede (CROSBY et al., 2016). Tem-se, ainda, o custo computacional e energético resultante do PoW.

2.4 SISTEMAS DE VOTAÇÃO BASEADOS EM BLOCKCHAIN

Como embasamento teórico para o desenvolvimento do protótipo foram analisados os sistemas *Follow my Vote* e *VoteWatcher*. Ambos utilizam *blockchains* como plataformas para armazenamento de votos em eleições, porém as abordagens adotadas pelos mesmos diferem bastante entre si. Esta seção descreve ambos os sistemas, elencando alguns de seus pontos chave.

2.4.1 Follow my Vote

O projeto *Follow my Vote*, proposto por uma startup de mesmo nome, propõe um sistema de votação onde os eleitores podem votar de seus próprios dispositivos pessoais, sem a necessidade de deslocamento até uma seção eleitoral. O sistema salva os votos em uma *blockchain* pública que pode ser auditada por qualquer dispositivo na rede (Follow My Vote, 2015).

O processo proposto pelo sistema inicia com o download da aplicação cliente no dispositivo do usuário. Durante a votação o eleitor provê através da aplicação três fotografias, sendo uma fotografia sua, uma de seu título eleitoral e a última de um documento oficial com foto. Um gestor trabalhando na votação verifica, remotamente, a validade das fotos providas e se o eleitor está apto a votar naquela eleição. Assim que a confirmação é concedida, o eleitor pode registrar seus votos. Este processo é análogo a um processo de votação tradicional, onde o eleitor apresenta estes documentos ao mesário em uma seção eleitoral (Follow My Vote, 2015).

Para garantir a anonimidade do eleitor, este não utiliza seu próprio título como identificador do voto. A aplicação gera um identificador para este eleitor usando um *token* que é assinado pela entidade verificadora a partir de um algoritmo de assinatura cega (do inglês: *blind signature*). Este processo garante que o voto não possa ser associado ao eleitor nem mesmo pela própria entidade verificadora. Como o eleitor conhece seu identificador, ele é capaz de verificar se os seus votos foram registrados de maneira correta (Follow my Vote, 2015).

O sistema do *Follow my Vote* permite, ainda, que o eleitor modifique seu voto depois de este já ter sido enviado. Neste cenário, um novo voto é divulgado para a rede e inserido em um novo bloco. Ao realizar a contagem, os *peers* consideram apenas o último voto inserido como sendo o válido. Todos os votos inseridos na rede são assinados através de um algoritmo de assinatura digital de curva elíptica (Follow my Vote, 2015).

A principal desvantagem da abordagem adotada pelo projeto é a possibilidade de coerção do eleitor ao voto, visto que o mesmo tem um mecanismo que prova que ele votou em determinado candidato. Outra consideração que deve ser analisada é a possibilidade de adulteração no processo de identificação, uma vez que qualquer agente que tenha em sua posse as três fotografias necessárias, consegue registrar o voto em nome do eleitor. Caso o dispositivo do eleitor seja comprometido por algum tipo de *malware*, o seu voto também pode ser comprometido.

2.4.2 VoteWatcher

O projeto *VoteWatcher*, implementado pela empresa *Blockchain Technologies Corporation*, propõe um sistema de votação baseado em urnas eletrônicas (utilizando urnas desenvolvidas pela própria empresa), onde os votos são armazenados em três *blockchains* diferentes, sendo uma destas local e as outras públicas. Durante o período da votação, os eleitores se dirigem até uma seção eleitoral e registram o seu voto em uma cédula de papel, que é posteriormente escaneada pela urna. A opção por manter a votação em papel permite a recontagem dos votos caso esta for necessária (Bitcoin Exchange Guide, 2018).

As urnas não são conectadas à Internet durante a votação, sendo os votos armazenados na *blockchain* local no decorrer deste período. A conexão e transmissão dos votos só é iniciada ao final das eleições. Os votos são então armazenados na *blockchain* da *Floricoin* e da *Bitcoin*, permitindo que qualquer usuário possa validá-los e realizar a contagem (Bitcoin Exchange Guide, 2018).

A abordagem adotada pelo *VoteWatcher* permite a realização da votação em um ambiente livre de coerção, em contraste com a abordagem adotada pelo *Follow my Vote*. A opção por conectar as urnas à Internet somente ao final do período de votação garante que o sistema não seja comprometido por ciberataques. Um sistema seguindo esta abordagem pode optar por associar o voto ao eleitor ou manter o voto totalmente anonimizado. Para a primeira opção é necessário que os votos sejam transmitidos em ordem aleatória ao final da eleição, visto que, a partir do momento do voto e da urna que o gerou, é possível deduzir a identidade do eleitor. Neste cenário, os votos também não podem conter informações relacionadas ao horário que foram gerados ou qualquer informação que revele a identidade do eleitor. A presença destas informações pode novamente introduzir a possibilidade de coerção. Em contrapartida, ao abrir mão da rastreabilidade do voto, o sistema se torna menos transparente, visto que o eleitor não tem como verificar se o seu voto foi registrado corretamente, restando-lhe confiar que a urna não fraudou o resultado.

3 DESCRIÇÃO DO PROTÓTIPO

A proposta deste trabalho é o desenvolvimento de um protótipo para um sistema de votação baseado em urnas eletrônicas onde os votos são armazenados em uma *blockchain* de acesso público, que pode ser auditada por qualquer pessoa, dificultando a manipulação dos resultados por atacantes ou pela entidade gestora do processo. Cada voto é assinado digitalmente pela urna que o gerou, permitindo aos auditores a verificação de sua proveniência. É computacionalmente impraticável para um *peer*, forjar votos ou blocos sem ter acesso à chave privada de uma urna válida. Um mecanismo de consenso baseado em PoW inviabiliza computacionalmente a alteração dos votos armazenados na *blockchain*, sendo requerido da urna (ou grupo de urnas) comprometida uma quantidade de poder computacional superior à das demais, conferindo aos dados armazenados um alto grau de imutabilidade. Tendo conhecimento do horário em que votou, o eleitor (ou qualquer indivíduo) pode verificar se o seu voto foi registrado corretamente pela urna, podendo ainda realizar a contagem dos votos em tempo real.

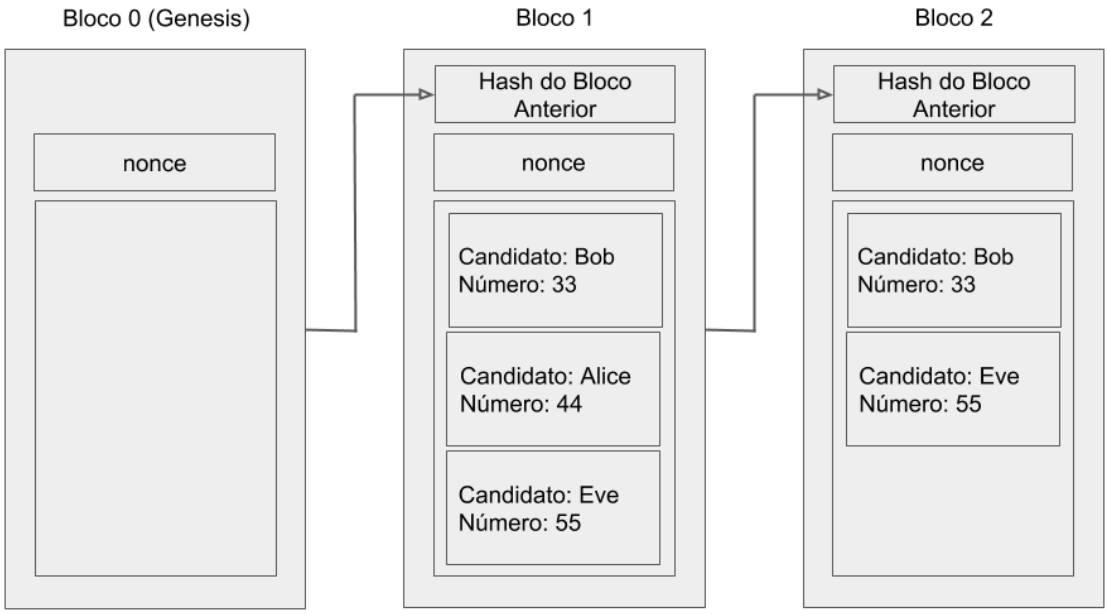
A Figura 3.1 apresenta uma versão simplificada da estrutura da *blockchain* utilizada para o armazenamento dos votos no sistema proposto. Similarmente à estrutura descrita na Figura 2.1, cada bloco na *blockchain* armazena a *hash* do bloco anterior, uma lista de transações e um valor *nonce* que valida o bloco perante ao PoW. Diferentemente da Figura 2.1, no entanto, ao invés de armazenar transações financeiras, a lista de transações contém os votos registrados na eleição. Cada bloco armazena os votos que estavam pendentes (não haviam sido inseridos na *blockchain*) no momento em que foi criado.

Ao longo deste capítulo o protótipo desenvolvido é descrito, discorrendo-se a respeito das decisões de design adotadas. A seção 3.1 descreve a arquitetura da aplicação e o racional por trás desta. Na seção 3.2 são discutidas as principais ferramentas utilizadas para a produção do algoritmo.

3.1 ARQUITETURA DO PROTÓTIPO

Embora seja possível desenvolver um sistema de votação totalmente *peer-to-peer*, sem qualquer intervenção de uma entidade centralizada, geralmente existe algum grupo que gerencia o processo eleitoral. É incumbida a este grupo a verificação da elegibilidade dos candidatos que concorrerão a cada cargo, analisando-se o cumprimento dos requisitos exigidos, como filiação a partido político e nacionalidade. Em um sistema inteiramente *peer-to-peer*, no entanto, esta atribuição incide sobre os próprios *peers* mantedores do processo, podendo ser adotada uma abordagem onde não há critério de exclusão para candidatos ou onde exista um protocolo de consenso que exija a aprovação do candidato

Figura 3.1 – Abstração da blockchain do sistema proposto.



Fonte: Produção do próprio autor.

por uma certa porcentagem dos *peers* (ou por um grupo de *peers* específico).

O software desenvolvido ao longo deste trabalho requer, como premissa, a existência de uma entidade gestora da eleição, não objetivando a integração do processo de triagem dos candidatos, o qual deve ser realizado em etapa prévia ao processo de votação pela própria entidade. Define-se, então, como primeira etapa da interação com o sistema o cadastro dos candidatos válidos. São incumbidos, ainda, à entidade gestora, o cadastro (descrito na seção 3.1.1) e a distribuição das urnas eleitorais contendo a aplicação instalada.

A integração do sistema com uma base de dados de eleitores poderia permitir a um atacante a obtenção de dados pessoais destes eleitores. O protótipo, portanto, não propõe a realização da identificação do eleitor e a verificação de que o mesmo está apto a votar, sendo estas atribuições incumbidas aos mesários no dia da eleição.

O sistema desenvolvido é composto por três aplicações:

- Autoridade (descrita na seção 3.1.1) - Provê aos *peers* conectados à *blockchain* um mecanismo para descoberta de outros *peers*, além de disponibilizar uma API para obtenção das chaves públicas das urnas válidas e dos dados dos candidatos em uma votação. Esta aplicação é provida de forma centralizada, sendo instalada em um servidor (ou conjunto de servidores) de conhecimento público;
- Auditor (descrita na seção 3.1.2) - Responsável pelo armazenamento e verificação

da autenticidade dos votos e da *blockchain*. Cada auditor expõe uma API que permite a sua comunicação com os demais *peers* conectados à rede. Qualquer usuário com acesso à rede da *blockchain* pode se registrar como um auditor da votação;

- Urna (descrita na seção 3.1.3) - Estende as funcionalidades de um *peer* auditor, sendo, ainda, responsável pela inserção de novos votos e blocos na *blockchain*. Para ser inserido como uma urna, o *peer* deve ser registrado previamente ao processo de votação, pela entidade que a gerencia.

Toda a comunicação entre as aplicações executando em diferentes *peers* é realizada utilizando o protocolo HTTP[1], que descreve um conjunto de métodos para transferência de recursos entre um servidor e um cliente. Conforme descrito no decorrer deste capítulo, a proveniência dos recursos transmitidos pela rede é garantida através de um algoritmo ECDSA, não sendo necessária a encriptação das informações trafegadas, visto que estas são de domínio público. A abordagem *peer-to-peer* confere a cada *peer* o status tanto de cliente (quando este requisita algum recurso de outro *peer*) quanto de servidor (quando responde a alguma requisição). Optou-se pelo protocolo HTTP pela grande quantidade de bibliotecas disponíveis para a maioria das linguagens de programação e por não manter uma sessão ativa entre os *peers* após a requisição ser completada. Para encontrar os recursos na rede o protocolo utiliza URLs. Cada aplicação expõe uma API para as demais, definindo as rotas e métodos HTTP utilizados para a requisição de recursos, além dos parâmetros que devem ser providos e do conteúdo da resposta.

3.1.1 Autoridade

Em um sistema computacional *peer-to-peer* deve existir um mecanismo que permita aos *peers* encontrar os demais na rede. Em decorrência de o sistema eleitoral proposto permitir que qualquer pessoa conecte-se à rede a qualquer momento da votação, faz-se necessário um sistema dinâmico, onde novos *peers* consigam obter a URL dos *peers* ativos e os já existentes consigam detectar a presença dos novos.

Como mecanismo para endereçar o cenário descrito, é necessária a existência de um ou múltiplos servidores centralizados, cuja URL seja conhecida por todos os *peers* da rede. Na ausência destes servidores é impossível para um *peer* descobrir a URL dos demais *peers* manutenedores da *blockchain*. A aplicação descrita nesta seção, nomeada como autoridade, executa sobre estes servidores centralizados, sendo seu gerenciamento realizado pela entidade promotora da eleição. Todos os *peers* possuem a URL da autoridade salva como uma constante no próprio código. Os auditores do processo eleitoral e

[1] IETF: https://tools.ietf.org/html/rfc2616

as urnas registram sua própria URL com a autoridade no momento em que se conectam à rede.

Por ser uma aplicação centralizada, é necessário que a entidade gestora adéque os servidores à quantidade de requisições esperada durante a eleição, empregando técnicas para garantir a integridade do serviço mesmo diante de ataques.

Foi incumbido à autoridade, ainda, prover uma lista de todos os candidatos válidos na eleição e uma lista das chaves públicas de todas as urnas válidas. Caso estas listas não sejam providas pela autoridade, ambas devem ser registradas no código fonte dos próprios *peers*, demandando uma revisão no código para cada votação. Ao utilizar-se a autoridade para esta finalidade, é possível que o software das urnas e dos auditores seja reutilizado em diversas eleições, sendo apenas requerido dos *peers* o download das listas quando estes se conectam à rede. Logo, demanda-se uma etapa prévia à eleição, onde a entidade deve cadastrar na aplicação autoridade todos os candidatos válidos e as chaves públicas das urnas eleitorais.

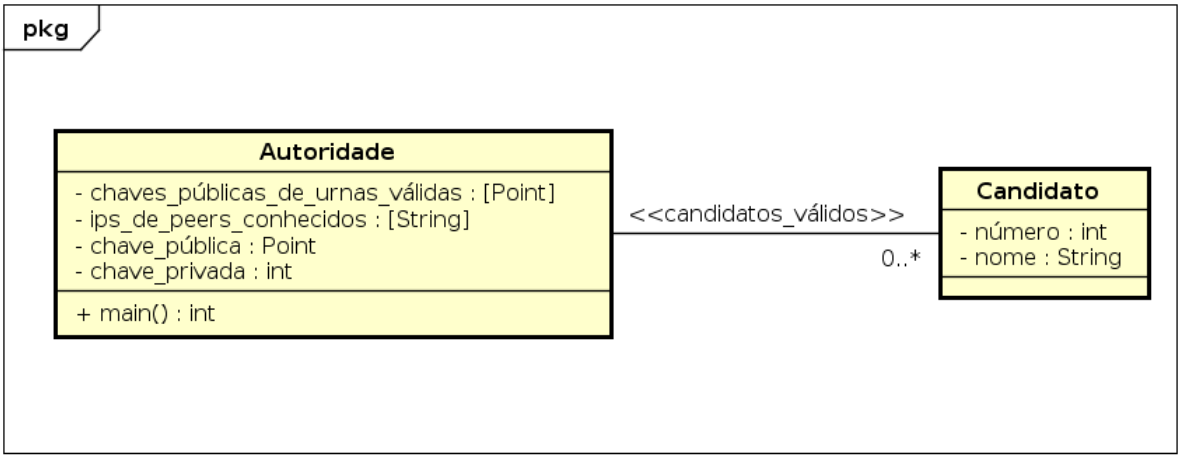
Requisições para a autoridade podem, no entanto, ser interceptadas por atacantes, que podem utilizar esta vulnerabilidade para enviar uma lista modificada de urnas ou candidatos válidos, levando o *peer* a acreditar que um voto é válido quando na verdade provém do próprio atacante. Como forma de garantir que os dados providos são realmente da autoridade, todos os recursos requisitados à autoridade são assinados utilizando um algoritmo de assinatura digital. Devido a maior segurança provida, utilizou-se um algoritmo de curva elíptica, baseado na curva *secp256k1*, que é definida pela equação $y^2 = x^3 + 7$. Esta curva foi padronizada em 1999 (QU, 1999) e é utilizada por diversas criptomoedas, como a *Bitcoin* e o *Ether*[2].

Quando a autoridade é instanciada, um par de chaves assimétricas é gerado utilizando-se um algoritmo de curva elíptica. A chave privada é utilizada para assinar os dados antes de enviá-los aos *peers*. A chave pública deve ser de conhecimento de todos os *peers* da rede, sendo integrada como uma constante nos seus códigos (junto à URL da autoridade). De posse desta chave, o *peer* pode determinar se a resposta que recebeu da requisição foi assinada pela chave privada da autoridade ou por um atacante. Caso o algoritmo determine que a assinatura e a chave pública da autoridade não são correspondentes, os *peers* apenas descartam o valor recebido.

O diagrama de classes da Figura 3.2, desenvolvido sob a especificação UML, apresenta a relação a nível estrutural entre as diferentes classes da aplicação autoridade. Observam-se na figura o par de chaves assimétricas utilizado pela autoridade, a lista de candidatos válidos e a lista contendo as urnas válidas.

[2]Bitcoin Wiki: https://en.bitcoin.it/wiki/Secp256k1

Figura 3.2 – Diagrama de classes estrutural da aplicação da autoridade.



Fonte: Produção do próprio autor.

3.1.2 Auditor

Tradicionalmente, o levantamento e contagem dos votos em eleições é incumbido a um grupo seleto de pessoas, tornando a adulteração dos resultados simples, caso este grupo entre em comum acordo em realizá-la. Da perspectiva dos eleitores, não existe garantia da integridade do processo. Para remediar este cenário, o sistema proposto divulga os votos em uma *blockchain* de domínio público, permitindo à qualquer eleitor verificar se o seu voto foi registrado corretamente, além de possibilitá-lo a realização da contagem dos votos e acompanhamento dos resultados em tempo real.

Para acessar a *blockchain*, o eleitor (ou qualquer pessoa que possua conexão com a rede da *blockchain*) deve instalar uma aplicação denominada auditor em algum dispositivo em sua posse. Esta aplicação recebe, em tempo real, todos os novos votos e blocos divulgados pelas urnas, validando-os e inserindo-os à sua cópia local da *blockchain*. Ao contrário da abordagem utilizada por criptomoedas como a *Bitcoin*, onde qualquer *peer* pode gerar transações financeiras e novos blocos, no sistema proposto somente urnas são capazes de gerar os votos e blocos a serem inseridos na *blockchain*.

Ao ser instanciado, o auditor envia uma requisição para a autoridade (cuja URL está registrada de maneira estática em seu código) solicitando a lista de candidatos válidos da eleição além da lista contendo a chave pública de todas as urnas válidas. A assinatura destas listas é, então, verificada, utilizando-se a chave pública da autoridade (também armazenada de maneira estática no código) para determinar sua proveniência. Caso a assinatura seja válida, as listas são salvas localmente. Em seguida, o auditor solicita à autoridade a lista de *peers* conhecidos, realizando novamente a validação da assinatura da resposta. Devido a novos *peers* poderem se conectar à rede a qualquer momento

durante a eleição, cada *peer* deve verificar constantemente se as suas cópias locais das listas estão atualizadas. Esta verificação é realizada a cada novo voto recebido da rede.

De posse da URL de todos os *peers* registrados pela autoridade, o auditor inicia o processo de download da *blockchain*. Para realizar este processo, o auditor envia requisições para cada *peer* conhecido até encontrar algum que possua uma cópia válida da *blockchain*. Isto implica que, para cada *peer*, o auditor realize o download e verificação completos de todos os votos e blocos inseridos na *blockchain*. Ao encontrar uma *blockchain* válida, o auditor passa a utilizá-la, mantendo-a atualizada com os novos blocos que recebe. Além da *blockchain*, o auditor requisita aos demais *peers* a lista de votos pendentes. Estes são votos que foram transmitidos pelas urnas, porém ainda não foram adicionados a nenhum bloco da *blockchain* (processo descrito na seção 3.1.3).

Ao longo do período de votação, diferentes *peers* na rede possuirão diferentes versões válidas da *blockchain*. Isto é causado por dois fatores:

- O tráfego dos blocos é dependente da velocidade de propagação dos dados pela rede. Quanto maior o tempo requerido para esta, maior será a diferença de informações entre os *peers*. Este fator é mitigado pelo PoW, porém deve-se levar em consideração que, em dado momento, *peers* diferentes podem apresentar dados divergentes, devido ao tempo de propagação da informação;

- Múltiplas urnas podem encontrar um *nonce* satisfatório para o PoW simultaneamente, realizando a divulgação de diferentes versões. Neste cenário, os *peers* adotarão a primeira versão válida da *blockchain* que receberem, dividindo a rede em grupos com *blockchains* distintas. Este fator é mitigado pelo próprio PoW. A próxima urna que criar um bloco com *nonce* válido passará a possuir a *blockchain* mais longa, que será adotada por todos os demais *peers*.

Ao final da votação, deve-se aguardar um período de tempo até todos os *peers* adotarem uma versão única da *blockchain*. O tamanho deste período é proporcional à dificuldade do algoritmo de PoW. Ressalta-se que, durante este período, as urnas continuarão gerando novos blocos vazios (não contendo nenhum voto). Estes blocos não possuem relevância na contagem dos votos.

Além das diferentes versões válidas da *blockchain*, atacantes podem manter versões alteradas. Apesar de estas versões serem descartadas pelos demais *peers*, esta técnica pode ser utilizada para ataques de negação de serviço (DoS, do inglês: *Denial of Service*), sobrecarregando os demais *peers* com a verificação de *blockchains* inválidas.

Ao receber novos votos de outros *peers*, cada auditor verifica sua proveniência e o candidato votado. Para ser considerado válido o candidato deve estar presente na lista de candidatos obtida da autoridade. É, portanto, impossível para qualquer atacante convencer os demais *peers* a aceitar um candidato inválido.

Para a validação da proveniência o auditor deve verificar se o voto foi gerado por uma urna válida. O sistema deve, ainda, prover resiliência caso a conexão com a urna for interceptada por um atacante. Como mecanismo para a verificação, cada voto é assinado pela urna que o gerou. O auditor executa, então, uma verificação para determinar se o voto foi assinado por uma chave pública válida. Para evitar que a verificação tenha que ser realizada para cada chave pública da lista obtida da autoridade, a urna inclui junto ao voto a sua própria chave pública, restando ao auditor verificar se a mesma está presente na lista de chaves válidas e se a assinatura é autêntica.

Para impedir que um atacante seja capaz de replicar um mesmo voto inúmeras vezes, cada voto deve possuir, além do candidato escolhido, um *timestamp* que o torna único para a urna na qual foi gerado. Este *timestamp* deve estar incluído no conteúdo assinado, impossibilitando a um atacante alterar este atributo sem invalidar a assinatura. Caso o voto passe pelo processo de validação ele é adicionado à lista de votos pendentes do auditor, que contém todos os votos que ainda não foram adicionados a nenhum bloco.

Além de votos individuais, o auditor ainda pode receber blocos. Blocos são criados por urnas válidas, e agregam um conjunto de votos que foram registrados durante um período de tempo (processo explicado na seção 3.1.3). Antes de ser adicionado à *blockchain* local do auditor, cada bloco deve ser validado.

Como somente urnas podem criar blocos novos para adicionar à *blockchain*, faz-se necessária, novamente, a utilização de assinaturas digitais para a garantia da proveniência. Cada bloco inclui, portanto, a chave pública da urna que o criou e a assinatura digital desta. Blocos que não provêm de urnas válidas são descartados.

Para a etapa seguinte do processo, o auditor busca no bloco o valor de seu índice. Três cenários são possíveis para o valor do índice:

- O índice é menor que o comprimento da *blockchain* local. Neste cenário o bloco pode ser descartado, visto que não agrega trabalho à *blockchain*;

- O índice é imediatamente subsequente ao último bloco da *blockchain* local. Neste cenário, o auditor busca no bloco o atributo que referencia a *hash* do bloco anterior. Este atributo é responsável por manter a ordenação dos blocos. Este valor é então comparado à *hash* do último bloco registrado em sua *blockchain* local, sendo adicionado ao final desta caso os valores sejam correspondentes. Caso o valor da *hash* do bloco anterior não seja equivalente ao último bloco registrado na *blockchain* local, é indicativo de que existe uma versão mais longa da *blockchain* diferente da versão atual do auditor. Neste cenário, é necessário que este requisite aos outros *peers* os últimos blocos recursivamente até que encontre o último bloco em comum entre a sua *blockchain* e a *blockchain* mais longa, verificando cada um destes blocos individualmente. Neste cenário, se a *blockchain* recebida for válida, o bloco passa a utilizá-la, descartando os blocos válidos de sua própria *blockchain* que são substituídos pela

versão maior, caso estes existam;

- O índice é maior que o comprimento da *blockchain*. Isto é indicativo de que o auditor não recebeu um ou mais blocos, sendo necessário requisitar estes blocos recursivamente aos demais *peers* até encontrar o último bloco em comum entre as duas *blockchains*. Neste cenário, novamente, o auditor deve verificar individualmente cada um dos blocos recebidos. Caso for determinada a sua validade, o auditor passa a utilizar a nova *blockchain*.

Após a verificação do índice, o auditor verifica a proveniência de cada voto incluído no bloco e a validade do candidato votado.

A próxima etapa da validação verifica a *hash* provida para aquele bloco. Para ser considerada válida, a *hash* deve iniciar por uma determinada quantidade de zeros. Esta quantidade é determinada pela dificuldade do algoritmo, que é definida estaticamente no código de todos os *peers*. Verifica-se, então, se o *nonce* provido para o bloco é válido. Para esta verificação, o bloco, o *nonce* e a *hash* do bloco anterior são utilizados como entrada para uma função *hash* criptográfica e a saída é comparada com a *hash* do bloco. Caso as duas não sejam correspondentes o bloco é descartado. É importante que a *hash* do bloco anterior seja um dos elementos utilizados como entrada para a função *hash*. Isto implica que a *hash* gerada para o bloco é dependente dos blocos anteriores e qualquer alteração em algum destes blocos invalidaria o cálculo do bloco atual.

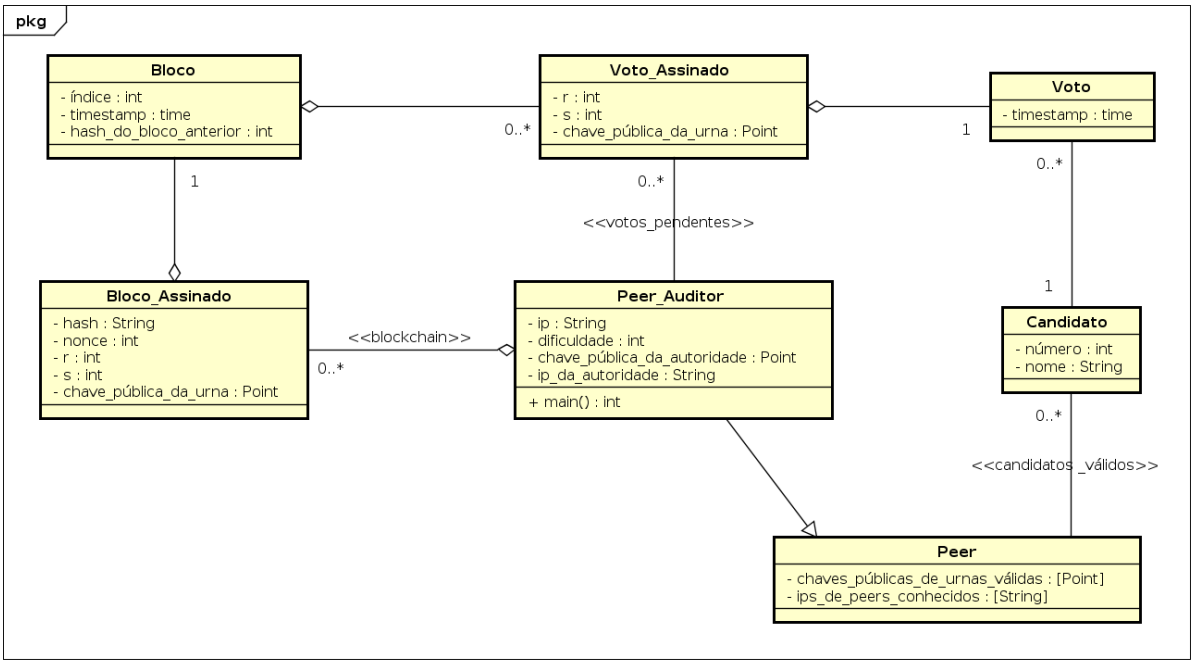
Caso o bloco seja validado em todas as etapas do processo, este é adicionado ao final da *blockchain*. O auditor, então, verifica sua lista de votos pendentes e remove todos os votos que estejam presentes no bloco adicionado.

As verificações da proveniência e integridade dos votos, blocos e *blockchains* recebidos de outros *peers*, impossibilitam que atacantes convençam os auditores a utilizar informações incorretas. Quaisquer informações recebidas que não provenham de urnas válidas ou votos registrados para candidatos inválidos são imediatamente descartados. A Figura 3.3 apresenta a relação estrutural entre as classes da aplicação auditor.

3.1.3 Urna

A aplicação urna é instalada sobre as urnas eletrônicas utilizadas durante a votação. Não existe limitação quanto ao seu uso em hardware específico. A aplicação pode, portanto, ser utilizada em qualquer plataforma que suporte as ferramentas utilizadas. O mesmo é válido para as demais aplicações propostas (autoridade e auditor). Esta característica permite a utilização do sistema em eleições de porte menor, onde não há recursos para a aquisição de dispositivos específicos para esta finalidade.

Figura 3.3 – Diagrama de classes estrutural da aplicação do auditor.



Fonte: Produção do próprio autor.

A aplicação urna estende as funcionalidades do auditor. Logo, assim como os auditores, as urnas realizam a verificação de todos os votos e blocos que lhe são enviados, somente adicionando à sua cópia local da *blockchain* os que forem aprovados em todos os testes de validação. A única funcionalidade dos *peers* auditores que não é provida por uma urna é o processo de download inicial da *blockchain*. Como as urnas operam a partir do início do período de votação, não é necessário que estas se comuniquem com os demais *peers* da rede para baixar uma *blockchain* válida, visto que a votação inicia com a *blockchain* não contendo nenhum voto.

Ao ser instanciada, a primeira incumbência da urna é a geração de um par de chaves assimétricas. A chave privada será utilizada constantemente ao longo de sua execução. A chave pública é registrada pela entidade gestora da eleição na aplicação da autoridade antes do período da votação, junto às chaves públicas das demais urnas, possibilitando que todos os *peers* na rede possam verificar a proveniência dos recursos recebidos.

Além das funcionalidades de um auditor, a urna é incumbida da criação de novos votos e blocos. O fluxo de execução de uma urna é composto por três *threads* principais, ambas iniciadas no momento que a urna é instanciada:

- A *thread* de comunicação com a rede é, essencialmente, igual ao processo do auditor, sendo responsável por lidar com as requisições dos demais *peers* na rede;

- A *thread* de registro de votos é o ponto principal de interação com o eleitor. Esta *thread* coleta o voto registrado através da interface da urna e o assina, sendo responsável por realizar o envio deste aos demais *peers*.

- A *thread* de cálculo da prova de trabalho é responsável por agregar os votos pendentes em um bloco e calcular um *nonce* que produza uma *hash* aceitável para a validação deste bloco. O cálculo do *nonce* ocorre constantemente, utilizando todo o poder computacional disponível da máquina. É imprescindível que todas as urnas possuam poder computacional equivalente, visto que urnas com maior poder computacional possuem uma probabilidade maior de resolver o PoW antes das demais.

A *thread* de registro do voto é implementada como um servidor http, porém, em contraste com a *thread* de comunicação, este servidor somente aceita requisições provenientes da própria máquina, ignorando requisições externas. Esta abordagem permite o isolamento entre a interface utilizada para interação com o usuário e o servidor de aplicação, tornando o sistema mais modular e, consequentemente, mais fácil de manter.

O registro de novos votos inicia na escolha do candidato pelo eleitor. O processo da interface então requisita à *thread* de registro que um novo voto para aquele candidato seja criado. Junto ao voto, a *thread* salva um *timestamp* e a assinatura digital, gerada a partir de sua chave privada, possibilitando aos demais *peers* a verificação de sua proveniência. Em seguida, a urna salva o voto criado em sua lista de votos pendentes e o envia para todos os *peers* conhecidos.

A função da *thread* de cálculo do PoW consiste em construir um bloco a partir da lista de votos pendentes. Este voto deve ainda agregar entre seus parâmetros a *hash* do último bloco inserido na *blockchain* da urna. Para garantir que este processo funcione desde o primeiro bloco, é necessário que exista um bloco inicial na *blockchain*, que é inserido estaticamente no momento em que cada *peer* é instanciado. Este bloco inicial não contém nenhum voto, sendo seu único propósito garantir que todos os *peers* possuam o primeiro bloco em comum.

Após ter o bloco criado, a *thread* passa a calcular o PoW, buscando encontrar um *nonce* que, ao ser utilizado como parâmetro para uma função *hash* criptográfica, produza uma *hash* que inicie com a quantidade de zeros especificada pela dificuldade do algoritmo. Junto ao *nonce*, a função *hash* recebe como parâmetro o próprio bloco criado, garantindo que o *nonce* encontrado não possa ser utilizado para blocos diferentes. Para a geração da *hash* optou-se pela utilização de um algoritmo *SHA-256*, que gera *hashes* de 256 bits. Este algoritmo, publicado em 2001, provê um nível de segurança superior aos algoritmos precedentes (GILBERT; HANDSCHUH, 2003), sendo utilizado pela *Bitcoin* desde sua introdução em 2009[3]. Ao ser encontrado um *nonce* satisfatório, a *thread* gera uma assinatura para o bloco criado utilizando a chave privada da urna. Em seguida, o bloco é adicionado ao

[3]Bitcoin Wiki: https://en.bitcoin.it/wiki/How_bitcoin_works

final da *blockchain* e os votos inseridos no bloco criado são removidos da lista de votos pendentes.

Como etapa final da iteração da *thread*, esta envia o bloco criado para todos os *peers* conhecidos, para que seja inserido em suas *blockhains*. Ao longo do tempo de execução desta iteração, que pode levar vários minutos dependendo da dificuldade escolhida, diversos novos votos podem ter sido registrados. A *thread* então lê a lista de votos pendentes e gera um novo bloco para o qual deve calcular o *nonce*. Este processo repete-se até o final da votação, estando todas as urnas em uma condição de corrida para determinar quem proverá o próximo bloco da cadeia.

Quando a *thread* de comunicação recebe um bloco válido de outro *peer*, esta deve ser capaz de instruir a *thread* do PoW a descartar o bloco que ela está calculando naquele momento. Este processo é realizado através de uma variável de controle, cujo valor é alterado logo após a *thread* de comunicação remover da lista de votos pendentes todos os votos que estejam presentes no novo bloco. A *thread* do PoW verifica constantemente o status da variável de controle para determinar se deve prosseguir com o cálculo do PoW ou encerrá-lo. Quando a segunda opção ocorre, o bloco sendo calculado é descartado e a *thread* cria um novo bloco utilizando a lista de votos pendentes.

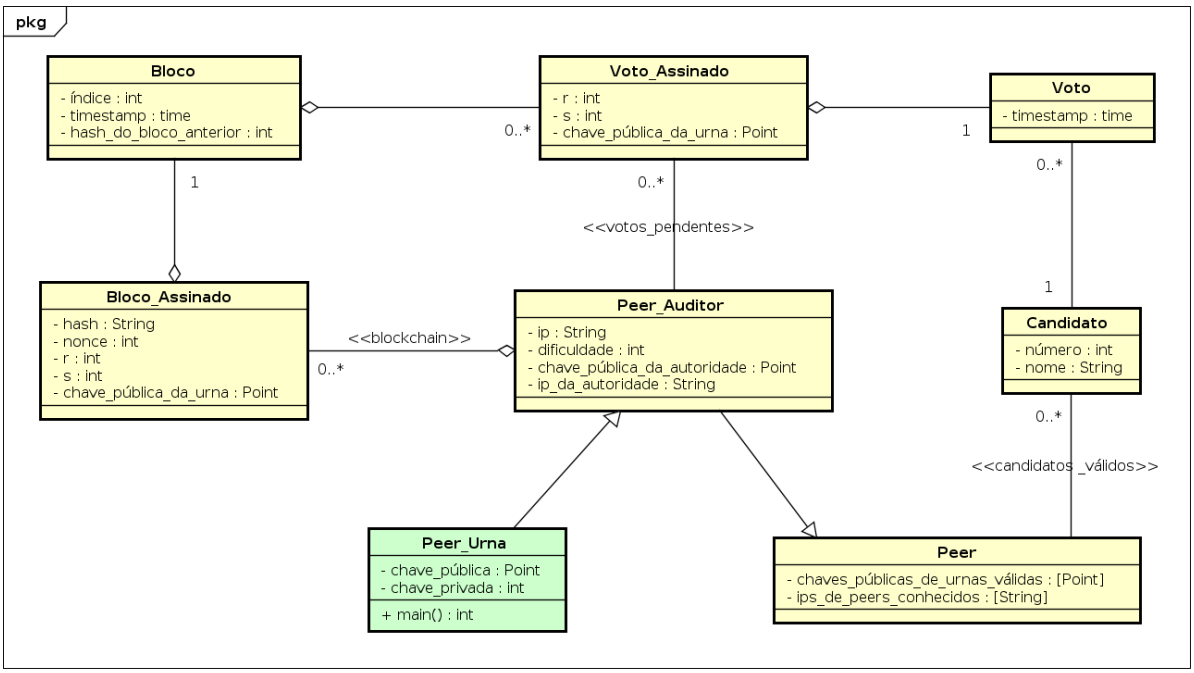
A utilização de múltiplas *threads* de execução demanda algumas considerações no algoritmo. *Threads* podem acessar o mesmo recurso simultaneamente, o que pode gerar leituras ou escritas inválidas sobre este recurso, caso uma troca de contexto entre as *threads* seja iniciada antes da operação ser finalizada. No algoritmo proposto são compartilhadas as regiões de memória correspondentes à lista de votos pendentes, à *blockchain* e às chaves criptográficas. Para evitar o corrompimento destas, é necessária a implementação de um mecanismo que garanta a exclusividade no seu acesso. Este mecanismo é implementado utilizando-se um *mutex* (acrônimo do inglês: *mutual exclusion*), que exige que cada *thread* solicite o recurso antes de poder utilizá-lo. No cenário em que uma *thread* solicita um recurso que estiver alocado para outra, a primeira abre mão de seu ciclo de processamento, aguardando até a segunda concluir a operação que está realizando.

A Figura 3.4 ilustra a relação estrutural entre as diferentes classes da aplicação urna. Observa-se a presença de todas as estruturas de auditores, com a adição da classe que confere a habilidade para a criação e assinatura de votos e blocos (em destaque).

3.2 FERRAMENTAS UTILIZADAS

Esta seção discorre sobre as principais ferramentas adotadas para a produção do algoritmo e o racional por trás de sua utilização. São apresentadas a linguagem de programação utilizada e as bibliotecas que proveem as funcionalidades para a criação do algoritmo.

Figura 3.4 – Diagrama de classes estrutural da aplicação da urna com enfoque na diferença em relação à aplicação do auditor.



Fonte: Produção do próprio autor.

Como linguagem de programação foi adotado o *Python*[4] (na versão 3.6.7). O *Python* possui interpretadores para inúmeras plataformas, incluindo microcontroladores, o que pode possibilitar a criação de dispositivos de votação mais econômicos (e possivelmente mais seguros). Além disso, a popularidade da linguagem entre programadores[5] torna provável a existência de bibliotecas que facilitem a implementação das funcionalidades do sistema, acelerando o processo de desenvolvimento. O desenvolvimento da linguagem de forma *open source* garante, ainda, que vulnerabilidades possam ser detectadas e corrigidas rapidamente pelos contribuidores do projeto.

Utilizou-se o *framework Flask* (na versão 1.0.2) por facilitar o processo de definição das rotas para lidar com requisições HTTP. O *framework* é ativamente mantido, sendo bastante popular entre desenvolvedores *Python*[6]. O *Flask* é responsável por analisar as requisições HTTP recebidas pelas aplicações e invocar as funções específicas que proveem as funcionalidades definidas pela API.

A biblioteca *Requests* (na versão 2.20.1) foi utilizada para a geração das requisições HTTP do lado do cliente. A biblioteca, desenvolvida em código aberto, é bastante popular entre desenvolvedores *Python*[7], facilitando a configuração de *headers* e a formatação do

[4]Python: https://www.python.org/doc/
[5]Tiobe Index:https://www.tiobe.com/tiobe-index/
[6]GitHub Flask: https://github.com/pallets/flask
[7]GitHub Requests: https://github.com/requests/requests

conteúdo a ser enviado.

Para a implementação do algoritmo para geração de *hashes SHA-256*, foi utilizada a biblioteca *hashlib*, que está entre as bibliotecas padrão do *Python*. Este fator lhe provê as mesmas vantagens já mencionadas para o uso da própria linguagem.

Para a geração das assinaturas digitais de curva elíptica, foi utilizada a biblioteca *fastecdsa* (na versão 1.6.5), desenvolvida por Anton Kueltz[8]. Esta biblioteca provê implementações para diversas curvas elípticas, contendo métodos para a geração de chaves assimétricas e para a geração e verificação de assinaturas digitais.

[8]GitHub fastecdsa: https://github.com/AntonKueltz/fastecdsa

4 DISCUSSÃO

A aplicação do sistema proposto em eleições deve levar em consideração uma série de fatores, que podem impactar tanto positivamente quanto negativamente o processo. Este capítulo analisa estes fatores, buscando determinar a viabilidade da utilização do sistema em votações reais.

O primeiro aspecto que deve ser considerado é a anonimidade do voto. O sistema proposto permite a qualquer pessoa determinar os candidatos em que um eleitor votou, caso tenha conhecimento do horário e da urna em que o voto foi realizado. Este fator pode levar candidatos ou grupos que potencialmente se beneficiem de determinados resultados a coagir eleitores a votarem de maneira diferente da qual votariam caso este fator não estivesse presente, utilizando desde a compra de votos até chantagens ou ameaças. Em sistemas de votação onde existe a completa anonimidade do candidato votado pelo eleitor, este fator é desestimulado por não haver uma garantia ao praticante da coerção de que o voto foi efetivamente registrado para o candidato específico.

A anonimidade é um dos principais fatores que devem ser analisados na adoção do sistema, pois invalida o seu uso em diversos cenários, como a eleição de governantes políticos. Em cenários onde coerção não é um fator determinante, no entanto, como votações cujo resultado não agrega benefícios significativos aos vencedores, a utilização do sistema é perfeitamente viável.

Em contrapartida, por abrir mão da anonimidade, o protótipo consegue prover um sistema de votação mais transparente, permitindo a auditoria de todo o processo por qualquer indivíduo. Este aspecto dificulta a adulteração do processo eleitoral, tanto por atacantes quanto pela entidade gestora da eleição, facilitando a detecção de fraude por qualquer *peer* na rede. Eleitores são capazes de verificar a integridade do registro de seu voto e qualquer *peer* é capaz de realizar a contagem em tempo real.

É computacionalmente inviável para qualquer atacante produzir um voto que seja aceito pelos demais *peers* como válido, a não ser que este obtenha a chave privada de alguma das urnas. Neste cenário, ainda, existem indicativos que podem ser utilizados pela rede para determinar a validade dos votos. A ordem de voto dos candidatos é geralmente mantida constante ao longo da eleição, então uma sequência de votos que não esteja ordenada corretamente pode ser considerada uma tentativa de inserção de votos inválidos. Ademais, geralmente existe um intervalo de tempo entre o último voto de um eleitor e o primeiro voto do próximo eleitor. Caso um atacante insira uma série de votos neste intervalo, a rede pode determinar que a diferença de tempo entre os votos é muito baixa, sendo insuficiente para a realização da troca de eleitores na urna. Um terceiro fator a considerar é a quantidade total de votos registrados ao final da votação, que deve igualar a quantidade de eleitores. Este fator pode ser mais difícil de detectar caso a entidade gestora

seja a responsável pela fraude, podendo divulgar uma quantidade maior de eleitores do que o número real.

O atacante pode ainda assumir o controle sobre uma urna, realizando a alteração dos próprios votos registrados pelos eleitores. Neste caso, os eleitores que votaram naquela urna poderão determinar a fraude a partir da consulta de seu próprio voto. Caso o atacante tenha efetivamente instalado algum software que lhe permita manipular os votos, este talvez possa ser detectado por um processo de auditoria do software da urna, ao final da eleição.

O protótipo desenvolvido demanda, ainda, a conexão constante das urnas a alguma rede compartilhada. Este fator torna o sistema extremamente vulnerável a ataques de negação de serviço. Um atacante pode, facilmente, sobrecarregar algum *peer* com requisições, forçando-o a processá-las enquanto os processos regulares do *peer* são postergados. Este fator é bastante agravante caso o *peer* alvo seja uma urna, visto que os novos votos registrados por ele serão salvos apenas localmente, sem serem transmitidos para os demais *peers*. Este cenário compromete o processo de validação dos votos em tempo real, porém, desde que o atacante não consiga assumir o controle sobre a urna e excluir a *blockchain* local, esta enviará os dados normalmente assim que o DoS for mitigado. Ressalta-se que, caso este cenário ocorra, os votos registrados na urna provavelmente serão inseridos em blocos na *blockchain* global somente a partir do momento que o ataque encerrar, uma vez que a quantidade de trabalho realizada pela urna atacada será inferior à do restante da rede, forçando-a a substituir os blocos locais que ela criou pelos blocos providos pelos *peers*.

Além dos cenários descritos, um atacante pode explorar outros vetores de ataque que são introduzidos pela utilização de redes de computadores no processo eleitoral. Indivíduos que pretendem utilizar a aplicação auditor para acompanhar as votações, devem, inicialmente, realizar o download desta. Este é o primeiro ponto que pode ser explorado por um atacante. Caso o atacante consiga iludir o usuário a baixar uma versão corrompida do sistema, o computador do usuário pode ser utilizado para a execução de ataques DoS distribuídos (DDoS, do inglês: *Distributed Denial of Service*) contra os demais *peers* da rede. A aplicação modificada do auditor, pode ser utilizada, ainda, para tentar convencê-lo de que a votação foi comprometida, reduzindo a confiança dos eleitores no sistema. Para reduzir o risco desta possibilidade, o download do software deve ser provido por um servidor certificado por uma autoridade de confiança do usuário, sob uma conexão encriptada que dificulte a interceptação do download. Atualizações no software podem ser assinadas digitalmente, permitindo ao auditor atestar a sua proveniência.

A exposição do sistema à rede permite, ainda, ao atacante a exploração de vulnerabilidades nos próprios dispositivos conectados, processo que seria muito mais complicado caso fosse demandada a inserção de uma mídia física diretamente na urna. Para reduzir o impacto destes riscos, o hardware e software das urnas deve ser validado exten-

sivamente, a fim de remover qualquer possibilidade de vulnerabilidade. A mitigação dos ataques DoS, no entanto, é mais complexa. É necessário que as urnas possuam acesso a uma quantidade de banda de rede bastante acima da requerida para seu tráfego normal, além de hardware e software que consigam lidar com grandes quantidades de requisições, garantindo à urna a capacidade de manter a operação de suas funcionalidades mesmo sob ataque. Devem-se, ainda, ser utilizadas ferramentas que consigam detectar padrões de ataque e tomar decisões sobre estes automaticamente, como o descarte de pacotes provenientes de determinados IPs (do inglês: *Internet Protocol*).

5 CONSIDERAÇÕES FINAIS

A atribuição do processo de armazenamento e auditoria dos votos em sistemas de votação aos próprios eleitores remove a necessidade de confiança na entidade gerenciadora da eleição. O protótipo desenvolvido ao longo deste trabalho, possibilita aos eleitores a verificação da integridade de todos os votos registrados pelas urnas eletrônicas e a contabilização dos resultados, criando um mecanismo incontestável de detecção de fraudes eleitorais e, consequentemente, aumentando a confiabilidade e transparência do sistema de votação. A utilização de assinaturas criptográficas digitais possibilita a verificação da proveniência dos votos, impedindo que usuários conectados à rede introduzam novos votos ou modifiquem os existentes, enquanto que o armazenamento destes votos em *blockchains* garante o consenso entre todos os seus mantenedores.

A divulgação dos votos sob domínio público, no entanto, acarreta na privação da anonimidade do eleitor, fator que pode fomentar a prática de coerção ao voto. Foram elencados alguns vetores de ataque que podem ser explorados por indivíduos ou grupos para buscar o comprometimento do protótipo desenvolvido, identificando-se algumas práticas que podem ser utilizadas nos sistemas que permeiam o protótipo para reduzir o impacto destes fatores.

Elencam-se quatro extensões que podem ser introduzidas ao protótipo em trabalhos futuros:

- Para o teste das funcionalidades do protótipo foi estipulado um valor arbitrário para a dificuldade do PoW. No entanto, em eleições reais, pode-se buscar uma dificuldade que garanta a inserção de novos blocos em intervalos de tempo determinados. A partir da quantidade de urnas disponíveis em uma eleição e no seu poder computacional é possível determinar-se a correlação entre a dificuldade do PoW e o tempo que a rede levaria para inserir novos blocos à *blockchain*. Elenca-se o estudo desta correlação como uma extensão deste trabalho;

- É proposto como trabalho futuro, a realização de testes de vulnerabilidade sistemáticos sobre os diferentes componentes do protótipo desenvolvido, a fim de determinar o impacto de cada vulnerabilidade e determinar estratégias que consigam mitigá-las ou reduzir seu impacto;

- O protótipo não é capaz de iniciar requisições HTTP com dispositivos por trás de *gateways* NAT (do inglês: *Network Address Translation*), sendo requerida a configuração de regras de redirecionamento de portas para estes casos. Uma extensão para o protótipo poderia ser proposta, analisando-se a utilização de técnicas de "*hole punching*"(pode ser traduzido grosseiramente do inglês como "furação") para estabelecer o canal de comunicação (FORD; SRISURESH; KEGEL, 2005), utilizando a

autoridade como intermédio;

- Extensões do protótipo podem propor a criação de clientes gráficos para cada uma das aplicações (autoridade, auditor e urna), facilitando a interação dos usuários com o sistema. O design destas aplicações como servidores HTTP torna o processo de construção de aplicações gráficas bastante modular, sendo apenas requerido o conhecimento das rotas definidas pela API.

O protótipo desenvolvido por este projeto remove a necessidade de confiança nos gestores da eleição, incumbindo a auditoria do processo de votação aos próprios eleitores, permitindo-lhes verificar a existência de fraudes neste processo em tempo real. Em contrapartida, no entanto, abre-se mão do sigilo do voto do eleitor, possibilitando a coerção deste por agentes mal-intencionados. Faz-se necessária para futuras pesquisas a busca por sistemas que consigam conciliar a anonimidade do eleitor com a transparência requerida para a confiabilidade do sistema eleitoral.

REFERÊNCIAS BIBLIOGRÁFICAS

ARANHA, D. et al. **Execução de código arbitrário na urna eletrônica brasileira**. 2018.

Bitcoin Exchange Guide. **VoteWatcher: Online Blockchain Voting & Election Services?** Bitcoin Exchange Guide, 2018. Acessado em nov 2018. Disponível em: <https://bitcoinexchangeguide.com/votewatcher/>.

CACHIN, C.; VUKOLIC, M. Blockchain consensus protocols in the wild. **CoRR**, abs/1707.01873, 2017. Disponível em: <http://arxiv.org/abs/1707.01873>.

CROSBY, M. et al. Blockchain technology: Beyond bitcoin. **Applied Innovation**, v. 2, p. 6–10, 2016.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. **IEEE transactions on Information Theory**, IEEE, v. 22, n. 6, p. 644–654, 1976.

Follow My Vote. **Blockchain Voting: The End To End Process**. Follow My Vote, 2015. Acessado em nov 2018. Disponível em: <https://followmyvote.com/blockchain-voting-the-end-to-end-process/>.

Follow my Vote. **Cryptographically Secure Voting**. Follow My Vote, 2015. Acessado em nov 2018. Disponível em: <https://followmyvote.com/cryptographically-secure-voting/>.

FORD, B.; SRISURESH, P.; KEGEL, D. Peer-to-peer communication across network address translators. In: **USENIX Annual Technical Conference, General Track**. [S.l.: s.n.], 2005. p. 179–192.

GILBERT, H.; HANDSCHUH, H. Security analysis of sha-256 and sisters. In: SPRINGER. **International workshop on selected areas in cryptography**. [S.l.], 2003. p. 175–193.

KATZ, J. et al. **Handbook of applied cryptography**. [S.l.]: CRC press, 1996.

LOWRY, S. Z.; VORA, P. L. Desirable properties of voting systems. In: **NIST E2E workshop**. [S.l.: s.n.], 2009.

MILLER, V. S. Use of elliptic curves in cryptography. In: SPRINGER. **Conference on the theory and application of cryptographic techniques**. [S.l.], 1985. p. 417–426.

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

OSGOOD, R. The future of democracy: Blockchain voting. **COMP116: Information Security**, 2016.

QU, M. Sec 2: Recommended elliptic curve domain parameters. **Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6**, Citeseer, 1999.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, ACM, v. 21, n. 2, p. 120–126, 1978.

ROGAWAY, P.; SHRIMPTON, T. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision

resistance. In: SPRINGER. **International workshop on fast software encryption**. [S.l.], 2004. p. 371–388.