UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A Privacidade e a Segurança de Dados no Uso de Blockchains em Urnas Eletrônicas: Analisando Modelos baseados em Ethereum

Rael Gugelmin Cunha

Trabalho de Conclusão de Curso - MBA em Gestão de Segurança de Dados (CEMEAI)

ICMC USP SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A Privacidade e a Segurança de Dados no Uso de Blockchains em Urnas Eletrônicas: Analisando Modelos baseados em Ethereum

Rael Gugelmin Cunha

USP - São Carlos

2022

RAEL GUGELMIN CUNHA

A Privacidade e a Segurança de Dados no Uso de Blockchains em Urnas Eletrônicas: Analisando Modelos baseados em Ethereum

Trabalho de conclusão de curso apresentado ao Centro de Ciências Matemáticas Aplicadas à Indústria do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, como parte dos requisitos para conclusão do MBA em Gestão de Segurança de Dados.

Área de concentração: Segurança de Dados

Orientador: Prof. Dr. Mario Gazziro

USP - São Carlos

2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

C972p

Cunha, Rael Gugelmin
A privacidade e a segurança de dados no uso de blockchains em urnas eletrônicas: analisando modelos baseados em Ethereum / Rael Gugelmin Cunha; orientador Mario Gazziro. -- São Carlos, 2022.
56 p.

Trabalho de conclusão de curso (MBA em Segurança de Dados) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Urnas eletrônicas. 2. Privacidade. 3. Blockchain. 4. Ethereum. I. Gazziro, Mario, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

FOLHA DE AVALIAÇÃO OU APROVAÇÃO

.

A minha esposa pela compreensão,
carinho e apoio incansável.

RESUMO

Cunha, Rael G. **A Privacidade e a Segurança de Dados no Uso de Blockchains em Urnas Eletrônicas:** Analisando Modelos baseados em Ethereum. 2022. 56 f. Trabalho de conclusão de curso (MBA em Gestão de Segurança de Dados) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Eleições transparentes constituem um dos grandes pilares de uma democracia sólida. E grande parte da responsabilidade desta transparência recai sobre o meio usado para que os cidadãos votem. Para agilizar o processo, alguns países, como o Brasil, adotam um modelo de urna eletrônica desde 1996. Recente alvo de críticas por não apresentar um modelo público e transparente de contagem e recontagem das apurações, eleitores que não votaram no candidato vencedor costumam colocar em dúvida o processo eleitoral (Teogenes, 2017). Com o crescente uso de tecnologias *blockchain* voltadas ao uso de programação geral (e não apenas focadas no uso financeiro), como a rede Ethereum, alguns modelos de sistema de voto eletrônico foram propostos nos últimos anos, propondo que as vantagens inerentes ao *blockchain* (imutabilidade, auditabilidade, descentralização) trariam a estes sistemas maior segurança, transparência, integridade e principalmente, tornariam fácil a tarefa de recontagem de votos. Neste trabalho, os modelos propostos sobre a rede Ethereum serão comparados entre si sob o foco da privacidade e segurança dos dados dos eleitores.

Palavras-chave: Urnas eletrônicas, privacidade, blockchain, Ethereum.

ABSTRACT

Cunha, Rael G. **Data Privacy and Security in the Use of Blockchains in Electronic Voting Machines:** Analyzing Ethereum-based Models. 2022. 56 f. Trabalho de conclusão de curso (MBA em Gestão de Segurança de Dados) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Transparent elections are one of the great pillars of a solid democracy. And much of the responsibility for this transparency falls on the means used to get citizens to vote. To speed up the process, some countries, such as Brazil, have adopted an electronic voting machine model since 1996. A recent target of criticism for not presenting a public and transparent model for counting and recounting the results, voters who did not vote for the winning candidate usually put in doubt the electoral process (Teogenes, 2017). With the increasing use of blockchain technologies aimed at general programming use (and not just focused on financial use), such as the Ethereum network, some models of electronic voting system have been proposed in recent years, claiming that the inherent advantages of blockchain (immutability , auditability, decentralization) would bring to these systems greater security, transparency, integrity and, above all, would make the task of recounting votes easier. In this work, the proposed models on the Ethereum network will be compared with each other under the focus of privacy and security of voter data.

Keywords: electronic voting machines, privacy, blockchain, Ethereum.

LISTA DE ABREVIATURAS E SIGLAS

DDoS	–	*Distributed Denial of Service*
P2P	–	*Peer to Peer*
PIN	–	*Personal Identification Number*
TSE	–	Tribunal Superior Eleitoral

SUMÁRIO

1 **INTRODUÇÃO**..... 11
1.1 **Motivação**..... 11
1.2 **Objetivos Gerais e Específicos**..... 12
1.3 **Estrutura do Documento**..... 12
2 **REVISÃO BIBLIOGRÁFICA**..... 13
2.1 **Blockchain**..... 13
2.2 **Bitcoin**..... 13
2.3 **Ethereum**..... 14
2.4 **E-Voting**..... 15
2.5 **Algoritmos de Prova de Conhecimento Zero**..... 15
2.6 **Artigos Relacionados**..... 15
3 **PROBLEMA ABORDADO**..... 18
3.1 **Descrição do Problema**..... 18
3.2 **Escopo**..... 19
4 **METODOLOGIA**..... 20
4.1 **Questões Levantadas**..... 20
4.1.1 **Questões Existenciais**..... 20
4.1.2 **Questões Descritivas**..... 20
4.1.3 **Questões de Causalidade**..... 21
4.2 **Busca de Artigos**..... 21
4.3 **Critério de Seleção de Artigos**..... 22
4.4 **Metodologia de Classificação**..... 22
5 **ANÁLISE E RESULTADOS**..... 24
5.1 **Análise**..... 24
5.2 **Resultados**..... 46
6 **CONCLUSÃO**..... 48
REFERÊNCIAS..... 49
GLOSSÁRIO..... 33
Apêndice A – Função votar do contrato AnonymousVoting.sol..... 35
APÊNDICE B – Apêndice B – Contrato Register..... 36

1 INTRODUÇÃO

1.1 Motivação

Embora o conceito de um livro razão eletrônico e imutável como o *blockchain* tenha sido inicialmente proposto por CHAUM (1979), foi apenas com a versão descentralizada proposta pelo *whitepaper* do Bitcoin (2008) que a tecnologia ganhou notoriedade.

Em pouco tempo, as vantagens do *blockchain*, como transações imutáveis, descentralização e dados validados por pares passaram a ser naturalmente vistas como aplicáveis não apenas a sistemas financeiros como o Bitcoin, mas aos mais variados sistemas que requisitem transparência, imutabilidade e auditabilidade, como urnas eletrônicas.

Sendo o Bitcoin, em sua versão atual, focado em sua criptomoeda, redes de *blockchain* mais generalistas surgiram, como o Ethereum, que possibilita que sistemas com finalidades diversas possam ser programados sobre essa rede.

Isso abriu a porta para modelos de voto eletrônico sobre a rede Ethereum surgissem.

Paralelamente, o Brasil adota um modelo de urnas eletrônicas desde 1996, que gradualmente foi ocupando o lugar das cédulas de papel, chegando ao ponto de hoje a maioria dos votos no país serem feitos por meio eletrônico (ARANHA et al, 2014), sendo o país com a maior eleição informatizada do mundo (TSE, 2022).

Embora a urna eletrônica brasileira historicamente sofra críticas quanto à sua transparência, no ano de 2021 esta discussão ganhou um foco muito maior, quando o presidente em exercício questionou a transparência e auditabilidade da mesma, chegando a propor que a contagem oficial fosse através de votos impressos pela urna, ao invés de contagem por registros eletrônicos.

O que vem de encontro aos modelos propostos sobre a rede Ethereum: estes trariam a integridade, segurança e auditabilidade de uma rede *blockchain*, eliminando de uma vez por todas as desconfianças de um sistema acusado de não ser transparente.

Mas será que os modelos propostos estão isentos de problemas? Estes modelos precisam atender a vários requisitos no que tange a privacidade do eleitor. Se os votos forem todos armazenados na *blockchain*, eles precisam evitar qualquer ligação ou mapeamento com um eleitor real, de forma a manter o voto secreto. Este sistema também deve permitir a totalização dos votos apenas depois de finalizada a votação. Também devem permitir que o eleitor seja capaz de verificar seu voto. E no caso, se cada eleitor precisar de uma carteira

digital do Ethereum para votar (como o Metamask), como ficaria a privacidade dos dados desta carteira?

Também é preciso lembrar que cada transação na rede Ethereum principal, assim como na rede Bitcoin, tem literalmente um preço a ser pago.

1.2 Objetivos Gerais e Específicos

O objetivo deste trabalho é, após descrever brevemente os conceitos utilizados (no Capítulo "**2 Revisão Bibliográfica**"), analisar os modelos de votação eletrônica propostos sobre o Ethereum utilizando critérios sistemáticos propostos por TAŞ e TANRIÖVER (2020), "A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting", que serão detalhados na seção "**4.4 Metodologia de Classificação**".

O segundo objetivo é analisar se, em vista dos critérios anteriores, algum dos sistemas propostos poderia ser um candidato viável a urna eletrônica brasileira em seu formato atual.

1.3 Estrutura do Documento

A apresentação deste trabalho está divida em 5 capítulos:

- Capítulo 1 (Introdução): Apresenta a movitação em desenvolver este trabalho e os objetivos buscados;
- Capítulo 2 (Revisão Bibliográfica): Faz uma revisão sobre conceitos envolvidos numa rede blockchain, além de revisar trabalhos anteriores publicados sobre votos eletrônicos em Ethereum;
- Capítulo 3 (Problema): Descreve o problema a ser atacado e qual o escopo deste trabalho;
- Capítulo 4 (Desenvolvimento): Descreve os critérios e analisa os modelos de votos eletrônicos em Ethereum propostos;
- Capítulo 5 (Conclusão): Apresenta as conclusões do trabalho, discutindo brevemente os resultados obtidos nesta monografia e sugestões para futuros trabalhos relacionados.

2 REVISÃO BIBLIOGRÁFICA

Nesta seção serão abordados os conceitos relacionados a este trabalho, como *blockchain*, Bitcoin, uma breve introdução a rede Ethereum, *e-voting*, como também serão apresentados alguns artigos acadêmicos relacionados ao tema.

2.1 Blockchain

Em CHAUM (1979) foi descrito um sistema computacional com foco no sistema bancário, cujos dados, uma vez validados por pares, seriam "selados" (ou seja imutáveis) em um mecanismo que ele chamou de "cofre", protegido por criptografia. Em HABER e STORNETTA (1991) foi descrito um mecanismo para proteger conteúdos digitais, através de blocos imutáveis encadeados e protegidos por criptografia. Com a ajuda de BAYER (1992), eles adicionaram o conceito de árvores Merkle ao sistema, que permitiu um ganho de tempo significativo na etapa de validação.

Em DAI (1998) propõe-se a recompensa com dinheiro a resolução de quebra- cabeças computacionais quando um consenso fosse atingido.

Em NAKAMOTO (2008), o *whitepaper* "Bitcoin: A Peer-to-Peer Electronic Cash System" (que cita os trabalhos anteriores em suas referências), acrescenta sobre os conceitos anteriores, fazer com que o sistema atue em uma rede distribuída em P2P.

Ou seja, em sua definição atual, uma rede *blockchain* (na maioria das vezes, distribuída), que guarda um histórico de transações armazenadas e imutáveis através de blocos encadeados, onde cada bloco guarda o *hash* do bloco anterior, até chegar o bloco original. Qualquer tentativa de alteração de histórico em algum dos blocos invalida toda a cadeia.

2.2 Bitcoin

O Bitcoin é uma moeda digital, sem lastro e sem um Banco Central ou qualquer outra autoridade bancária como operador, e que usa como sistema computacional a sua própria rede *blockchain* desenvolvida em 2008 por um autor (ou grupo de pessoas) desconhecido, sob a alcunha de Satoshi Nakamoto (NAKAMOTO, 2008). Sua implementação se tornou código aberto. Cada transação (que contém uma origem e um destino para um determinado valor) aguarda em uma fila, até ser validada por um nó especial chamado de minerador. Cada minerador é recompensado pela validação de uma transação com uma pequena quantidade de

moedas digitais (*bitcoins*). E quanto mais transações validar, maior sua classificação dentro da rede, o que o tornará um nó com maior preferência a receber novas validações (sistema chamado de Prova de Trabalho). Após serem validadas, as transações são armazenadas, conforme já citado anteriormente, em blocos interligados através do *hash* do bloco anterior.

2.3 Ethereum

Com a popularização do Bitcoin como investimento, as vantagens do uso de *blockchain* (transparência, imutabilidade e integridade), se tornaram alvo de possíveis usos além de sistemas financeiros. Por que não utilizar *blockchains* em qualquer área de atuação onde as vantagens citadas acima sejam requisitos básicos? Em alguns setores, o uso de *blockchains* pareceu então se encaixar como uma luva: registros imobiliários (CONSENSYS, 2022), vendas de propriedades não fungíveis (como obras de arte, por exemplo), ou ainda sistemas de voto eletrônico, o foco deste trabalho.

Algumas tentativas de tentar usar a própria *blockchain* do Bitcoin foram feitas, mas como seu foco principal é sua criptomoeda, e sua linguagem de *scripting* (a UTXO) apresenta algumas limitações significativas: não guardar estado, não ser Turing completa, dentre outras deficiências (BUTERIN, 2014), começaram a surgir outras implementações *blockchain* como focos específicos, como Namecoin (NAMECOIN, 2022) para registros de nomes, ou a Colored Coins (COLORED COINS, 2022) que permite as pessoas criar suas próprias moedas digitais, mas sem que nenhuma dessas implementações ganhasse a mesma popularidade do Bitcoin.

Em 2014, Vitalik Buterin (BUTERIN, 2014) descreveu um sistema *blockchain*, chamado de Ethereum, cujo principal foco seriam as aplicações computacionais rodando sobre essa rede. As aplicações foram denominadas de *smart contracts*, e cada transação, assim como no Bitcoin, também teria um custo computacional envolvido, medido em unidade de chamada *gas*. Esta unidade tem um custo em uma moeda própria, chamada de *ether*, que deve ser comprada pelos usuários para poderem operar.

Ou seja, o Ethereum em si consiste de uma máquina virtual distribuída, rodando *smart contracts* desenvolvidos em uma linguagem própria para isso, o que acabou lhe concedendo a alcunha de *blockchain* 2.0, por se tratar de uma evolução sobre o sistema utilizado no Bitcoin.

Sua relativa flexibilidade permita que sejam criadas as mais variadas aplicações, desde novos tipos de moedas digitais, vendas de artigos não fungíveis, até mesmo sistemas de voto eletrônico, o foco deste trabalho.

2.4 E-Voting

Voto eletrônico é o processo de votação feito com o auxílio de um dispositivo eletrônico em substituição a uma cédula de papel. Geralmente é dividido em duas categorias (WIKIPEDIA, 2022): voto eletrônico que requer uma urna eletrônica (que não necessariamente usa a Internet); e o voto feito através de um dispositivo qualquer com acesso a Internet (como computadores pessoais ou *smartphones*).

A urna brasileira cai na primeira categoria (embora os votos sejam enviados dos TREs através de VPNs para os servidores do TSE que contabilizarão os totais).

Um sistema de voto construído sobre a rede Ethereum cai na segunda categoria: boa parte das implementações é construída visando o navegador de Internet como interface.

2.5 Algoritmos de Prova de Conhecimento Zero

Em criptografia, Provas de Conhecimento Zero são algoritmos que permitem provar que "alguns teoremas são válidos sem fornecer nenhuma pista do porquê" (BLUM, FELDMAN e MICALI, 1988]. Ou seja, uma das partes consegue provar efetivamente que possui determinado conhecimento, sem relevar as técnicas ou qualquer outra informação utilizada para tal.

2.6 Artigos Relacionados

Antes de citar os artigos relacionados, é importante fazer uma menção ao trabalho de ARANHA et al (2014), que fizeram parte da Segunda Edição do Teste Público de Segurança do Sistema Eletrônico de Votação e compartilharam alguns detalhes e deficiências da urna eletrônica utilizada no Brasil.

KOÇ et al (2018) em "Towards Secure E-Voting Using Ethereum Blockchain" demonstraram como funcionaria um sistema simples de e-voting usando Ethereum, com um sistema que checa se o voto está no intervalo de tempo permitido e também não permite que o eleitor vote repetidamente. Mas citam a inerente falta de privacidade deste modelo: nada é feito para esconder o hash do usuário ou ainda para quem ele votou e quando, dados que ficarão armazenados na blockchain. Também não há preocupação com a totalização prematura dos votos.

KHOURY et al (2018) em “Decentralized Voting Platform Based on Ethereum Blockchain” propõe um modelo em que os eleitores se registram no sistema de através de um aplicativo para celular, e que automatiza o registro na plataforma Ethereum e cobra de cada eleitor o custo em *gas* de seu registro. Neste modelo, o eleitor deve ser registrado previamente através do uso de SMS e posteriormente vota através de um aplicativo para celulares. Existe a checagem de voto prévio. Mas também não há proteção com a exposição de dados (como hora do voto) nem com a totalização prematura dos votos.

FAYEMI, THOMPSON e AYENI (2021) em “Design and Implementation of Electronic Voting Using KECCAK256 Algorithm on Ethereum Network” propõe um modelo similar, onde a preocupação é em ocultar a identificação do eleitor. Neste caso, a identificação do eleitor é armazenada em um sistema externo (com informações governamentais), através de um algoritmo KECCAK256, que segundo os autores, “seria muito difícil de ser interceptado por cyber-terroristas”. O eleitor utiliza uma interface web para votar. Infelizmente o artigo não fornece nenhum código dos *smart contracts* em seu texto e também não fornece link para acessar o código fonte, tornando impossível sua análise sob os aspectos propostos neste trabalho.

CHRISTYONO, WIDJAJA e WICKSANA (2021) em “Go-Ethereum for electronic voting system using clique as proof-of-authority” criam um sistema de votação sobre o Ethereum (utilizando a linguagem Go) com o objetivo de recriar a eleição para representante de conselho regional de 2019 na região de Jelupang, na Indonésia, com dados oficiais obtidos através da página da Comissão Geral de Eleição da Indonésia. O sistema não requer sistemas ou redes de terceiros, e cada eleitor deve-ser registrar primeiro na rede Ethereum. Embora contenha os mesmo problemas de privacidade apontados nos dois artigos acima, ele utiliza uma implementação de *Proof of Authority* (PoA), que seria uma alternativa proposta ao *Proof of Work* (PoW) ainda utilizado pelo Ethereum até a presente data, onde a validação de cada transação, no modelo PoA, se daria apenas por nós reconhecidos e selecionados previamente como autoridade.

McCORRY, SHAHANDASHTI e HAO (2017) propuseram, em “A Smart Contract for Boardroom Voting with Maximum Voter Privacy” um modelo e implementação de voto eletrônico sobre o Ethereum chamado Open Vote Network. Segundo os autores, um dos grandes problemas das soluções existentes até então era usar a blockchain apenas como armazenagem de votos e ter que confiar a contagem de votos a entidades externas. A entidade externa poderia ser não confiável ou comprometida, o que não traria a transparência e integridade requerida a um processo eleitoral. Segundo eles, a solução proposta reside

inteiramente no Ethereum, sem a necessidade de contagem de votos externa. O sistema de votos mostrado como exemplo, contempla apenas uma votação binária (opções “sim” e “não”), mas um exemplo de como pode ser tratada uma eleição com mais opções é mostrado. Além disso, este sistema propõe tratar a privacidade do eleitor e seu voto através da obfuscação da identidade do eleitor e de seu voto através do uso de algoritmos de zero conhecimento prévio. Também leva em conta problemas como mostrar totais apenas após o final da votação, e problemas inerentes dos smart contracts, como outros contratos que poderiam usar suas funções para impersonificar um eleitor.

E, por último, o trabalho de TAŞ e TANRIÖVER (2020), “A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting” faz uma análise sistemática sobre desafios e oportunidades do *e-voting* em *blockchain*, fornecendo critérios que serão usados neste trabalho para classificar os sistemas em Ethereum analisados.

3 PROBLEMA ABORDADO

3.1 Descrição do Problema

Após a popularização dos sistemas baseados em *blockchain* para o uso em criptomoedas, seu uso tem se extendido para além do sistema financeiro, devido às vantagens inerentes desse tipo de sistema: descentralização, integridade e imutabilidade. Ou seja, vantagens que parecem se encaixar perfeitamente com um sistema de votação eletrônica.

Algumas tentativas comerciais foram colocadas em prática, como as eleições presidenciais de Serra Leoa (TECHCRUNCH, 2018), e o uso de um aplicativo de votação (chamado Voatz) usado pelo estado de Utah na eleição presidencial norte-americana (GOVERNMENT TECHNOLOGY, 2020). Em ambos os casos, as soluções são proprietárias, e as empresas alegam disponibilizar pericias para os interessados.

Ou seja, no caso acima, embora a tecnologia *blockchain* seja usada, as soluções compartilham algumas das críticas feitas sobre urna eletrônica brasileira: falta de transparência do processo de votação e apuração, segurança por obscuridade, inspeção apenas sob demanda, desconhecimento da armazenagem dos dados. Também não fica claro se as soluções proprietárias usam *blockchains* descentralizadas ou não, o que permitiria um ponto único de falha.

No meio acadêmico, alguns artigos propõem soluções que utilizam a rede *Ethereum*, o que traria vantagens sobre os modelos comerciais acima: todo *smart contract* rodando sobre o Ethereum é de código aberto; os dados armazenados são imutáveis e podem ser facilmente inspecionados por qualquer um; o sistema é descentralizado, tornando muito mais difícil um ataque do tipo *DDoS* ser bem sucedido.

O objetivo deste trabalho é analisar algumas das soluções propostas, comparando-as através de critérios sistemáticos, no que tange a privacidade dos dados dos eleitores e a segurança do processo eleitoral e apuração, já que embora uma rede *blockchain* como o Ethereum tenha inúmeras vantagens, não deixa de trazer consigo pontos negativos: toda transação sempre traz consigo o *hash* identificador de origem e destino, o dado que será armazenado na transação e a data e hora desta.

Ou seja, se os votos forem simplesmente salvos publicamente na *blockchain* na hora em que são entrados pelo eleitor, seria possível para um observador já ir contando os totais, além de tentar fazer uma análise comportamental dos eleitores com base na hora de votação.

Além disso, muitos usuários (que no caso seriam os eleitores) acabam em algum momento expondo suas *hashes* identificadoras, o que tornaria fácil saber em quem votaram.

Pessoas que fazem uso frequente do Ethereum como meio de pagamento, também poderiam ter seu comportamento analisado. Por exemplo: alguém que receba doações constantes em valores fixos, poderia ser vasculhado nos sistemas de *log* online pelo valor constante e hora e data das doações, tendo assim seu *hash* de identificação associado a sua pessoa, o que permitiria saber em que candidato votou.

Além disso, o fato dos totais poderem ser calculados a qualquer hora teria grande influência no resultado da eleição: eleitores poderiam mudar seu voto ou simplesmente desistir de votar (ou anulando seus votos) ao verem que seu candidato está com baixa contagem de votos em determinado momento.

Embora redes *blochains* com foco em privacidade estejam aparecendo, as chamadas *privacy coins* (como Monero e ZCash) não permitem a rastreabilidade dos blocos registrados. Algumas *blockchains* generalistas, como a Cardano, apostam em pseudo-anonimização: mantém todos os dados rastreáveis, exceto origem e destino da transação.

Mas com o uso de algumas técnicas de anonimização de dados do eleitor, alguns modelos prometem manter as vantagens do Ethereum sem comprometer a privacidade do eleitor ou o resultado da eleição.

3.2 Escopo

Este trabalho analisará os artigos relacionados na revisão bibliográfica, com base na privacidade dos dados e segurança do processo de votação, e quando disponível, uma breve análise do repositório de código fonte.

Não é foco deste trabalho analisar a segurança de cada um dos processos criptográficos envolvidos.

4 METODOLOGIA

Este capítulo apresenta a metodologia utilizada no desenvolvimento deste trabalho. A Seção 4.1 apresenta as questões levantadas antes de se iniciar a busca por artigos. A Seção 4.2 apresenta qual o critério usado para buscar artigos científicos relacionados. A Seção 4.3 descreve quais os critérios foram utilizados para selecionar os artigos usados neste trabalho. E na Seção 4.4, são descritos quais os métodos de avaliação que os modelos de *e-voting* sobre o Ethereum serão classificados.

4.1 **Questões Levantadas**

A busca por artigos teve como objetivo responder às seguintes questões:

4.1.1 Questões Existenciais

Exploratória: existem urnas eletrônicas em blockchain (ethereum)? Existe literatura sobre a privacidade delas?

Descritiva/Classificação: quais os tipos de urnas eletrônicas em blockchain? Que tipo de dados de usuários elas armazenam?

Descritivo-Comparativas: uma urna eletrônica em blockchain oferece transparência e privacidade superiores a uma urna eletrônica "tradicional" (usada no Brasil, por exemplo)?

4.1.2 Questões Descritivas

Frequência/Distribuição: existe registrado algum uso de urnas eletrônicas em blockchain? Caso sim, qual a frequência?

Descritivas de processos: existe algum código fonte disponível? Como foi feito o processo de votação?

Relacionamento: no caso de existir alguma eleição registrada utilizando Ethereum, a eleição funcionou bem? O resultado foi respeitado pelos candidatos? Alguém apontou algum ponto negativo no processo, quando comparado a uma eleição "tradicional"?

4.1.3 Questões de Causalidade

Causalidade: O uso de *blockchain* diminui as reclamações sobre os resultados?

Causalidade Comparativa: O uso de *blockchain* melhora a privacidade quando comparado à urna tradicional?

4.2 Busca de Artigos

Num primeiro momento, para responder a questão descritiva de frequência e distribuição (“existe registrado algum uso de urnas eletrônicas em *blockchain*?”) foram feitas buscas utilizando mecanismos de buscas “tradicionais” como o Google, em busca de notícias sobre algum possível uso registrado de eleições que tenham ocorrido utilizando votação eletrônica em *blockchain*.

Num segundo momento, foram feitas buscas sobre implementações de *e-voting* em Ethereum. Utilizando como buscador primário o *Google Scholar*, *Sucupira Qualis* e *Impact Factor Journal*, foram feitas buscas alternando entre as seguintes chaves: ethereum, *e-voting*, *blockchain*.

Também foram analisadas as referências bibliográficas de alguns dos artigos. Curiosamente o trabalho de McCORRY, SHAHANDASHTI e HAO (2017), “A Smart Contract for Boardroom Voting with Maximum Voter Privacy”, além de cronologicamente ser um dos mais antigos e ser um dos únicos que parece adereçar os problemas de privacidade inerentes ao *e-voting* e ao Ethereum, é citado apenas em “Towards Secure E-Voting Using Ethereum Blockchain” (KOÇ et al, 2018), como um dos trabalhos analisados, mas descartados “porque seu protocolo inclui operações matemáticas complexas e então, requer vasto poder computacional, não sendo, então, amigável a Internet das Coisas”.

E ainda, numa terceira etapa, foram buscados artigos que fornecessem critérios para analisar os sistemas de *e-voting*. Poucos resultados válidos foram encontrados, mas o artigo “A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting” (TAŞ e

TANRIÖVER, 2020), faz uma análise sistemática que forneceu a base dos critérios que serão usados para classificar os modelos propostos.

4.3 Critério de Seleção de Artigos

Os artigos retornados foram ordenados por citação, e depois descartados (ou mantidos) pelo seguinte critério:

- Devem ser específicos para o Ethereum;
- Devem no mínimo ter algum trecho de código fonte disponível (quando não um repositório);
- Computa apenas um voto por eleitor;
- Devem ter a preocupação mínima com a privacidade do eleitor (ao menos escondendo a identidade do mesmo, ou no mínimo, reconhecendo ter essa fragilidade).

Outros critérios de seleção chegaram a ser cogitados, como:

- O sistema previne a totalização antes do final da votação?
- O sistema leva em conta o custo das transações?
- O sistema previne que seu *smart contract* seja usado maliciosamente por outros para "*impersonar*" um eleitor?
- O sistema leva em conta o poder computacional envolvido em grandes eleições?
- O sistema realmente se baseia na rede principal do Ethereum ou utiliza apenas as versões de teste?

Mas boa parte dos artigos não se preocupou em cobrir ou discutir estes critérios, como será visto adiante.

4.4 Metodologia de Classificação

TAŞ e TANRIÖVER (2020) propõem em “A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting” que todo sistema de *e-voting* deva incluir as seguintes características:

1. Não produzir nenhum recibo de votação para o eleitor, para que não possa ser comprovado o voto para um candidato em particular;
2. Não produzir resultados preliminares, de forma a não afetar a decisão de outros eleitores;
3. Integridade de dados: uma vez salvos, os dados não podem ser alterados;
4. Privacidade/votação anônima: a identidade dos eleitores e para quem votaram devem ser preservadas;
5. Somente eleitores registrados podem votar;
6. Eleitores podem votar somente uma única vez;
7. Robustez: o sistema deve operar de forma a não perder nenhum voto, no sentido de escalabilidade (ou seja, aguentar o tráfego);
8. Confiabilidade: o sistema deve operar de forma a não perder nenhum voto devido a *bugs* existentes.
9. Verificabilidade: o sistema deve permitir que os eleitores verifiquem que seus votos foram contabilizados de maneira correta.

Serão sob os critérios acima que os trabalhos propostos serão analisados.

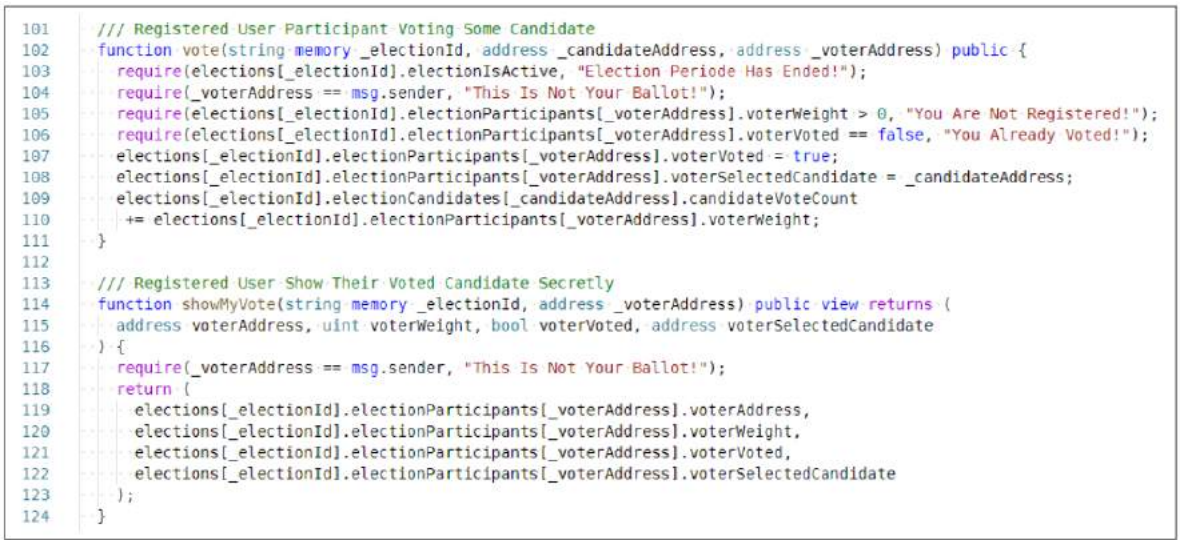
5 ANÁLISE E RESULTADOS

Ao analisar os critérios descritos anteriormente, serão utilizados os códigos fontes disponíveis. No caso de alguma parte não existir, será levado em conta a explicação provida pelos autores.

Em “Towards Secure E-Voting Using Ethereum Blockchain” (KOÇ et al, 2018) o código fonte das funções relevantes são incluídas no próprio texto.

Por sua vez, em “Decentralized Voting Platform Based on Ethereum Blockchain” (KHOURY et al, 2018), o código fonte está contido no artigo como parte do apêndice.

Já em “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021), apenas um trecho de código é incluído como figura:



```
101  /// Registered User Participant Voting Some Candidate
102  function vote(string memory _electionId, address _candidateAddress, address _voterAddress) public {
103    require(elections[_electionId].electionIsActive, "Election Periode Has Ended!");
104    require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
105    require(elections[_electionId].electionParticipants[_voterAddress].voterWeight > 0, "You Are Not Registered!");
106    require(elections[_electionId].electionParticipants[_voterAddress].voterVoted == false, "You Already Voted!");
107    elections[_electionId].electionParticipants[_voterAddress].voterVoted = true;
108    elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate = _candidateAddress;
109    elections[_electionId].electionCandidates[_candidateAddress].candidateVoteCount
110      += elections[_electionId].electionParticipants[_voterAddress].voterWeight;
111  }
112
113  /// Registered User Show Their Voted Candidate Secretly
114  function showMyVote(string memory _electionId, address _voterAddress) public view returns (
115    address voterAddress, uint voterWeight, bool voterVoted, address voterSelectedCandidate
116  ) {
117    require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
118    return (
119      elections[_electionId].electionParticipants[_voterAddress].voterAddress,
120      elections[_electionId].electionParticipants[_voterAddress].voterWeight,
121      elections[_electionId].electionParticipants[_voterAddress].voterVoted,
122      elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate
123    );
124  }
```

Figura 1: Código fonte de “Go-Ethereum for electronic voting system using clique as proof-of-authority”

E, finalmente, em “A Smart Contract for Boardroom Voting with Maximum Voter Privacy” (McCORRY, SHAHANDASHTI e HAO, 2017) é disponibilizado um repositório Github (McCORRY e JOHNSON, 2017) com o código fonte.

5.1 Análise

Os quatro trabalhos terão seu código fonte analisado sob a ótica dos dez critérios estabelecidos no capítulo anterior.

5.1.1 Não produzir recibo

O primeiro critério de análise, "Não produzir nenhum recibo de votação para o eleitor", implica que nenhum dos trabalhos avaliados produza um recibo apontando que candidato foi votado por determinado eleitor.

Quanto ao recibo impresso, não confundir com urnas eletrônicas que utilizam um voto impresso (sem identificação do eleitor) como segundo método de apuração: o critério aqui é verificar se não é gerado um recibo onde haja alguma ligação entre o eleitor e o candidato escolhido.

Geralmente isso implica num recibo impresso, mas também poderia se referir, no caso de um sistema computacional, a uma chamada de *log*, por exemplo.

No caso de *blockchains*, ainda pode também se referir ao sistema salvar a transação de voto identificando o eleitor e o candidato escolhido, mas esse cenário específico será analisado na seção 5.1.4 Privacidade/votação anônima.

Ao analisar o código responsável pela votação em cada trabalho, temos em "Towards Secure E-Voting Using Ethereum Blockchain" (KOÇ et al, 2018) o objeto eleitor modelado conforme a *struct* abaixo:

```
struct Voter {
    bool isVoted;
    bool hasRightToVote;
    uint8 vote;
    address ID;
}
```

E segundo os autores, a variável *vote* "armazena a escolha do eleitor entre todos os candidados". A função de votar, por sua vez, é mostrada abaixo (o identificador candidato escolhido é passado como parâmetro):

```
function vote(uint8 toProposal) public {
    Voter storage sender = voters[msg.sender];
    if (sender.isVoted || toProposal >= proposals.length &&
!sender.hasRightToVote)
        return;
    sender.isVoted = true;
    sender.vote = toProposal;
    proposals[toProposal].voteCount += 1;
```

```
    }
```

Neste caso, pode-se ver que não há nenhuma chamada para funções de impressão ou *logs*.

Em “Decentralized Voting Platform Based on Ethereum Blockchain” (KHOURY et al, 2018) a função para votar é mostrada abaixo:

```
function VoteFor(bytes32 candidate) public payable {
    if ((verify[msg.sender] == true) || (completed == true)
|| (msg.value == 0) || !isEligble(msg.sender))
        revert();
    else {
        votes[candidate] += 1;
        totalNumVotes += 1;
        verify[msg.sender] = true;
    }
}
```

Novamente, é possível ver que não há chamadas para funções de impressão ou *logs*.

Em “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021), a função de voto é mostrada como a seguir:

```
101  /// Registered User Participant Voting Some Candidate
102  function vote(string memory _electionId, address _candidateAddress, address _voterAddress) public {
103      require(elections[_electionId].electionIsActive, "Election Periode Has Ended!");
104      require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
105      require(elections[_electionId].electionParticipants[_voterAddress].voterWeight > 0, "You Are Not Registered!");
106      require(elections[_electionId].electionParticipants[_voterAddress].voterVoted == false, "You Already Voted!");
107      elections[_electionId].electionParticipants[_voterAddress].voterVoted = true;
108      elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate = _candidateAddress;
109      elections[_electionId].electionCandidates[_candidateAddress].candidateVoteCount
110          += elections[_electionId].electionParticipants[_voterAddress].voterWeight;
111  }
112
```

Também, sem chamadas para funções de impressão ou *logs*, conforme pede o requisito.

E, por sua vez, “A Smart Contract for Boardroom Voting with Maximum Voter Privacy” (McCORRY, SHAHANDASHTI e HAO, 2017) disponibiliza o código da função de votar no contrato `AnonymousVoting.sol`), que pode ser vista no Apêndice A. Novamente, mais um trabalho aprovado no requisito não produzir recibo: não há chamadas para funções de impressão ou *logs* gerados.

5.1.2 Não produzir resultados preliminares

Mostrar resultados parciais de uma eleição durante o processo de votação pode levar a dois problemas principais (McCORRY, SHAHANDASHTI e HAO, 2017):

- comportamento adaptativo: o eleitor, influenciado pelo resultado parcial, resolve votar em um candidato que não pretendia;

- comportamento abortivo: o eleitor, influenciado pelo resultado parcial desfavorável ao seu candidato, resolve não votar mais.

Em "Towards Secure E-Voting Using Ethereum Blockchain" (KOÇ et al, 2018), os autores fornecem uma função para leitura do candidato com mais votos:

```
     function winningProposal() public constant returns (uint256
_winningProposal) {
          uint256 winningVoteCount = 2;
          _winningProposal=0;
          for (uint8 prop = 0; prop < proposals.length;
     prop++)
               if (proposals[prop].voteCount > winningVoteCount)
               {
                    winningVoteCount = proposals[prop].voteCount;
                  _winningProposal = prop;
               }
     }
```

Conforme o código acima, a função simplesmente percorre o *array* de votos por candidato e retorna qual deles tem mais votos. Não há nenhum tipo de verificação.

Em "Decentralized Voting Platform Based on Ethereum Blockchain" (KHOURY et al, 2018), além de uma função que retorna o número total de votos, existe uma função de contagem de votos por candidato:

```
     function getTotalVotesFor(bytes32 candidate) public view
returns(uint) {
          return (votes[candidate]);
     }
```

Neste caso, também nenhuma verificação é feita.

Em "Go-Ethereum for electronic voting system using clique as proof-of-authority" (CHRISTYONO, WIDJAJA e WICKSANA, 2021), não é mostrado nenhum código para contagem de votos, mas baseado na função de votos, é possível ver que eles são armazenados na seguinte estrutura:

```
    elections[_electionId].electionCandidates[_candidateAddress].ca
ndidateVoteCount
```

Então é possível supor que uma função de contagem deva simplesmente percorrer o *array* de candidatos e ver qual deles possui o maior número de votos. Como a função de contagem e totalização não existe e os autores também não se pronunciam a esse respeito em seu trabalho, não é possível afirmar qual era a intenção deles nesse sentido. Ou seja, se eles realmente se preocuparam ou não com totalização parcial antes do final da eleição.

Em "A Smart Contract for Boardroom Voting with Maximum Voter Privacy" (McCORRY, SHAHANDASHTI e HAO, 2017), a totalização é feita pela função `votesTally`, que pode ser vista na linha 1000 do contrato `AnonymousVoting.sol` (abaixo apenas as primeiras linhas, relevantes para a discussão, são mostradas):

```
        function computeTally() inState(State.VOTE) onlyOwner {

           uint[3] memory temp;
           uint[2] memory vote;
           uint refund;

           // Sum all votes
           for(uint i=0; i<totalregistered; i++) {

               // Confirm all votes have been cast...
               if(!votecast[voters[i].addr]) {
                  throw;
               }

               vote = voters[i].vote;

               if(i==0) {
                 temp[0] = vote[0];
                 temp[1] = vote[1];
                 temp[2] = 1;
               } else {
                   Secp256k1._addMixedM(temp, vote);
               }
           }

           // All votes have been accounted for...
           // Get tally, and change state to 'Finished'
           state = State.FINISHED;
```

Na primeira linha, é possível ver que há uma checagem de estado: só é possível invocar a função se o sistema estiver no estágio de votação.

No laço `for` que faz a contagem, é possível também ver que a primeira checagem é para ver se todos os eleitores registrados votaram. Caso contrário, o sistema chama `throw`, que disparam uma exceção e interrompe a execução da função.

E a função também altera o estado do sistema para `FINISHED`, o que finaliza o processo de votação.

Os autores dizem que devido ao método que utilizam para calcular os votos totais (busca exaustiva), é possível que o último eleitor a votar possa simular os totais, o que permitiria que o mesmo alterasse seu voto ou abandonasse a eleição. Para isso, introduziram uma etapa anterior de "compromisso" (que o administrador da eleição pode opcionalmente requerer), onde antes da rodada de envio de seus votos para o *blockchain*, os eleitores enviam a sua intenção: o voto é anonimizado através de um *hash*, e então salvo para posterior comparação, para evitar que o eleitor mude seu voto na hora de enviar a transação.

5.1.3 Integridade de Dados

Como todos os sistemas são construídos sobre o Ethereum, ou seja, sobre um sistema de *blockchain* distribuído e grande o suficiente para prevenir ataques por força bruta, uma vez que os votos estejam gravados em transações aprovadas, eles estarão imutáveis e podem ter sua integridade verificada por terceiros.

Claro que isso é verdade desde que os *smart contracts* estejam rodando na rede principal do Ethereum, e não em seus múltiplas redes de teste: estas não fornecem garantia alguma sobre seus dados serem preservados; e no passado, já foram reiniciadas, devido a transição (atualização) de protocolos ou ataques mal intencionados.

Os trabalhos "Towards Secure E-Voting Using Ethereum Blockchain" (KOÇ et al, 2018), "Decentralized Voting Platform Based on Ethereum Blockchain" (KHOURY et al, 2018) afirmaram usarem as redes de teste (respectivamente Rinkeby e Ropsten) apenas como forma de evitar os custos reais associados as transações feitas na rede Ethereum principal. McCORRY, SHAHANDASHTI e HAO (2017) afirmam ter utilizado a "rede de teste principal", o que na época da escrita de seu trabalho era a rede Ropsten, também como forma de evitar os custos reais da rede principal.

Mas o trabalho "Go-Ethereum for electronic voting system using clique as proof-of-authority" (CHRISTYONO, WIDJAJA e WICKSANA, 2021) utiliza uma rede privada cuja validação é a Prova de Autoridade (PoA) ao invés da Prova de Trabalho utilizada na rede principal. Na Prova de Autoridade, os nós validadores recebem uma reputação mais alta e são

escolhidos arbitrariamente, geralmente por um grupo de entidades considerado confiável. Os autores justificam sua escolha pelo fato de que, segundo seu trabalho, redes *blockchains* que utlizam PoA são mais seguras contra ataques de colisão. Segundo a biblioteca usada pelos autores para fornecer a rede privada (Geth), a implementação de rede privada desta biblioteca usando PoA é bem mais eficiente (em termos de tempo e recursos computacionais) do que a versão que usa PoW.

O principal problema disso é que este trabalho fica restrito às redes privadas ou redes de teste, já que a rede principal do Ethereum não prevê utilizar PoA (atualmente usa *Proof of Work* e planeja mudar para *Proof of Stake* no final de 2022). Ou seja, no cenário de rede privada, a descentralização é perdida. No cenário de redes de testes, não há garantia de que os dados continuarão a existir, ou que não sofram ataques (DDoS e outros) como essas redes já sofreram no passado.

5.1.4 Privacidade/votação anônima

O sistema não pode permitir se associar, de forma alguma, o voto do eleitor com o candidato escolhido. Neste caso iremos analisar se o sistema, quando o eleitor efetua um voto, não armazena de forma direta no blockchain sua identificação e o candidato escolhido, de forma a não comprometer a privacidade do voto.

Não é o suficiente supor que o eleitor possuir um *hash* de conta na rede Ethereum garanta sua privacidade: este registro que permite associar o eleitor ao seu *hash* estará armazenado em algum sistema externo. Dessa forma, qualquer comprometimento desse sistema externo, ou qualquer exposição do *hash* por parte do eleitor, já elimina a privacidade da sua identidade.

Ou seja, esperar que o *hash* do eleitor seja o suficiente para torná-lo anônimo é uma suposição inválida: seria o mesmo que assumir que se a urna eletrônica expusesse dados de voto com o candidato escolhido amarrados ao título de eleitor fosse o suficiente para proteger a privacidade.

Em “Towards Secure E-Voting Using Ethereum Blockchain” (KOÇ et al, 2018), conforme já visto anteriormente, o voto é armazenado no atributo `vote` da *struct* `Voter`, ou seja, em `Voter.vote`. A função de votar, por sua vez, é mostrada abaixo (o identificador candidato escolhido é passado como parâmetro):

```
    function vote(uint8 toProposal) public {
```

```
        Voter storage sender = voters[msg.sender];
        if (sender.isVoted || toProposal >= proposals.length &&
!sender.hasRightToVote)
            return;
        sender.isVoted = true;
        sender.vote = toProposal;
        proposals[toProposal].voteCount += 1;
    }
```

Os autores ainda descrevem que os eleitores serão identificados por sua carteira e serão eles que estarão executando a função de votar (ou seja, o eleitor será o *sender* no código acima).

Na linha `sender.vote = toProposal`, podemos ver que o voto é armazenado guardando em `Voter.vote` o número do candidato escolhido. Depois, a chamada para essa função é salva no *blockchain*, conforme a Figura 1 abaixo mostrada pelos autores:

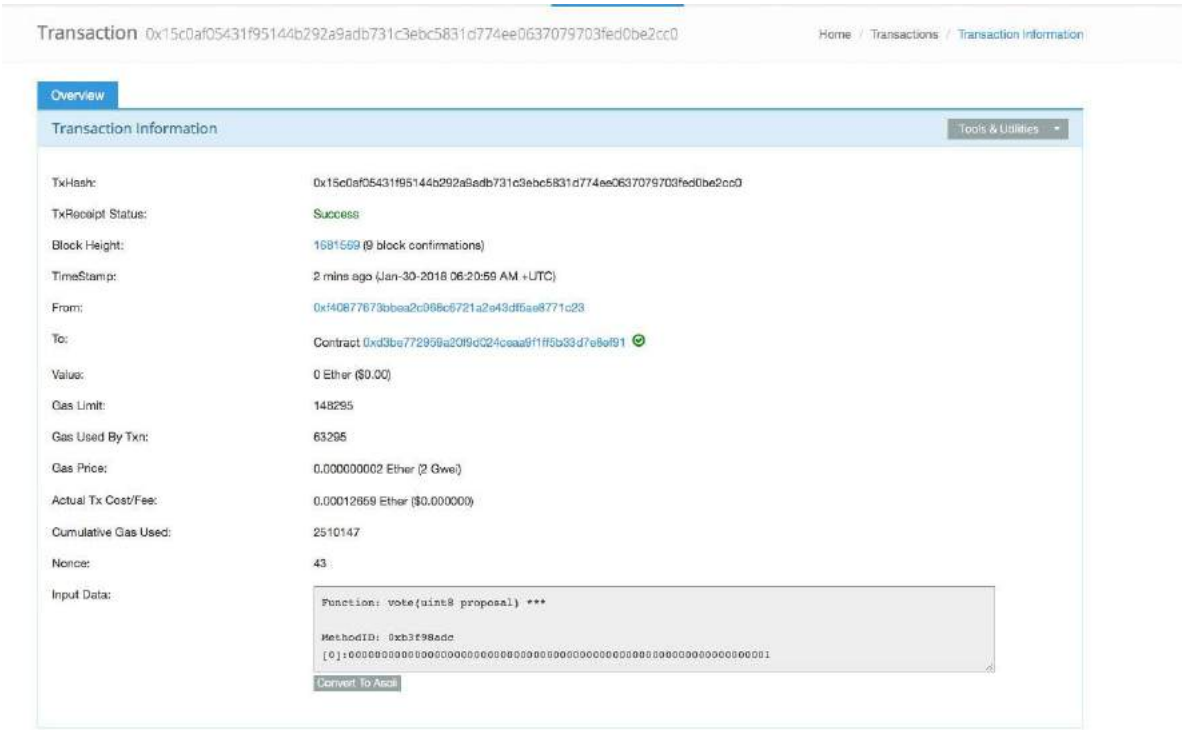


Figura 2: Captura de tela da transação fornecida pelos autores

Usando o id de transação acima, é possível consultar a transação acima e ver qual o candidato votado e quando (parâmetro *proposal* na seção *Input Data*, na parte inferior):

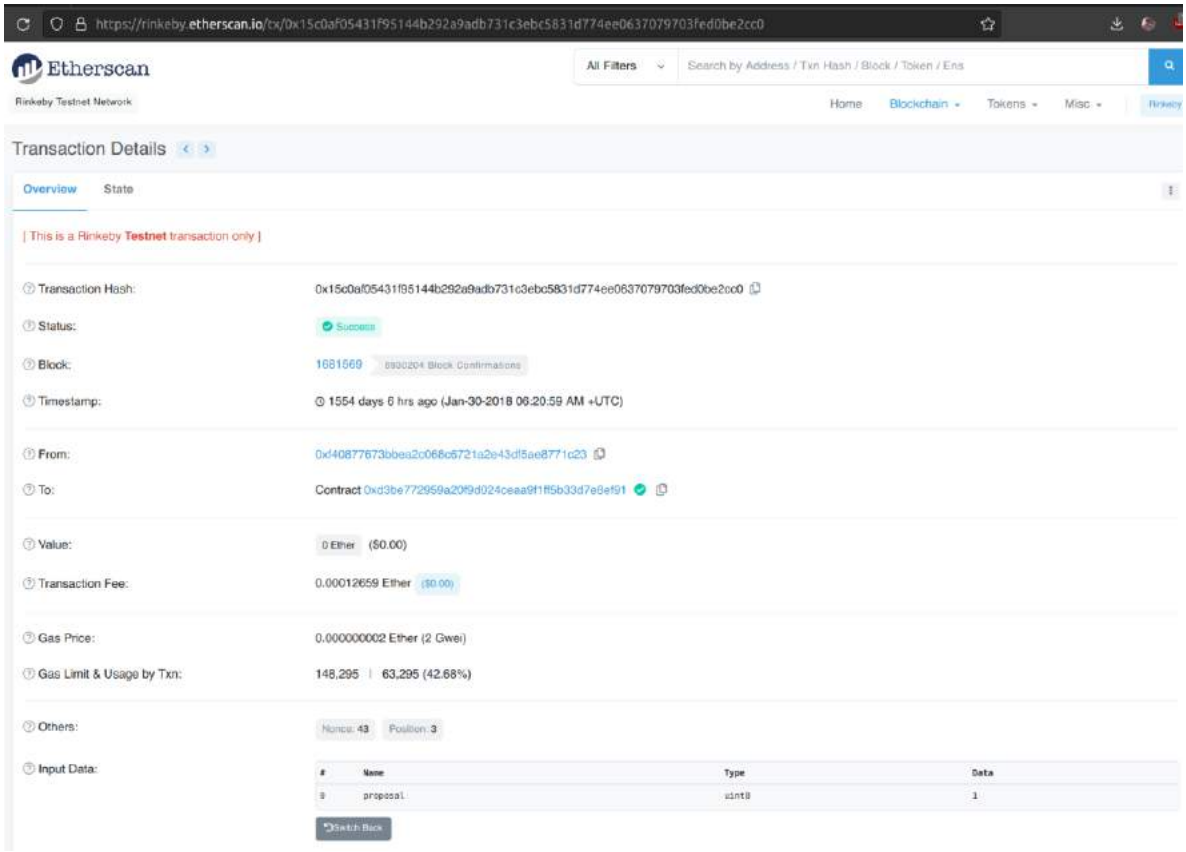


Figura 3: Consulta da transação no Etherscan, com o candidato de id 1 votado.

Ou seja, neste critério o modelo acima proposto expõe os dados do eleitor e o candidato escolhido.

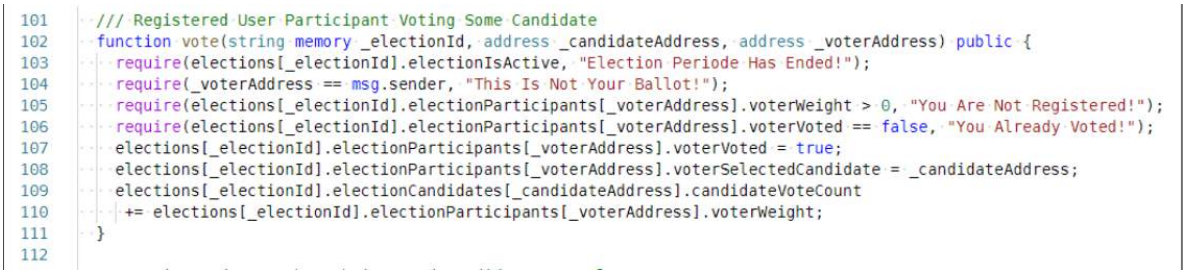
No próximo trabalho, "Decentralized Voting Platform Based on Ethereum Blockchain" (KHOURY et al, 2018), a função para votar é mostrada abaixo:

```
function VoteFor(bytes32 candidate) public payable {
    if ((verify[msg.sender] == true) || (completed == true)
|| (msg.value == 0) || !isEligble(msg.sender))
        revert();
    else {
        votes[candidate] += 1;
        totalNumVotes += 1;
        verify[msg.sender] = true;
    }
}
```

Pode-se notar que, em comparação ao modelo anterior, o voto para um determinado candidato não é armazenado em uma estrutura da memória. Mas o problema do registro da transação na *blockchain* continua acontecendo: os autores afirmam que o eleitor é quem irá executar o *smart contract* de votação (`VotingContract`). Ou seja, novamente basta analisar

o registro de transações do Ethereum para saber quem foi o `sender` e qual o valor passado para o parâmetro `candidate`.

Em “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021), a função de voto é mostrada como a seguir:



```
101  /// Registered User Participant Voting Some Candidate
102  function vote(string memory _electionId, address _candidateAddress, address _voterAddress) public {
103      require(elections[_electionId].electionIsActive, "Election Periode Has Ended!");
104      require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
105      require(elections[_electionId].electionParticipants[_voterAddress].voterWeight > 0, "You Are Not Registered!");
106      require(elections[_electionId].electionParticipants[_voterAddress].voterVoted == false, "You Already Voted!");
107      elections[_electionId].electionParticipants[_voterAddress].voterVoted = true;
108      elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate = _candidateAddress;
109      elections[_electionId].electionCandidates[_candidateAddress].candidateVoteCount
110          += elections[_electionId].electionParticipants[_voterAddress].voterWeight;
111  }
112
```

Segundo os autores, um único *smart contract* é o responsável por registrar os eleitores e votar. Uma vez que um dos eleitores inicia um processo de votação, ele é impedido de votar, e ficará responsável por registrar o voto de outros eleitores.

Na chamada da função `vote` podemos ver que o identificador da eleição, o endereço Ethereum do candidato e o endereço Ethereum do eleitor são passados como parâmetros. Como o código completo não está disponível, pressupõe-se que, como nos trabalhos anteriores, o processo de voto seja armazenado em uma transação. Ou seja, também seria trivial ver que carteira Ethereum votou em cada candidato. Além disso, na linha 108 é possível ver que é armazenado em memória o voto de cada eleitor e em qual candidato ele votou.

E, finalmente, em “A Smart Contract for Boardroom Voting with Maximum Voter Privacy” (McCORRY, SHAHANDASHTI e HAO, 2017), o processo de votação é antecedido por uma etapa onde é feita uma geração de um conjunto de chaves pública/privada, através de um aplicativo Java, `votingcodes.java`. Estas chaves são salvas em um arquivo de texto. O sistema então escolhe aleatoriamente um par de chaves para o eleitor votante e valida a chave de outros eleitores através de um algoritmo de zero conhecimento prévio (ou seja, valida sem identificar que eleitor está associado à determinada chave).

Só após todos obterem suas chaves de votação anônimas e validarem as chaves de outros eleitores, é que acontecerá a etapa de votação. Durante essa etapa, o sistema usa as chaves fornecidas, de forma a não identificar diretamente o eleitor. E depois o sistema fornece outra prova de zero conhecimento prévio de que o eleitor votou em uma das opções válidas.

Apenas após essa validação, e de haver encerrado o prazo de votação, é que o sistema registra a transação na rede Ethereum, através da função `submitVote` do arquivo `AnonymousVoting.sol`. Um exemplo de chamada da função pode ser visto na linha 350 da interface *web* `vote.html`:

```
anonymousvotingAddr.submitVote.sendTransaction(params, {
        from: web3.eth.accounts[accounts_index],
        gas: 4200000
    }
);
```

Onde `params` conterá o voto cifrado. Desta forma, quando a transação for salva na *blockchain*, o candidato escolhido não poderá ser identificado.

5.1.5 Somente eleitores registrados podem votar

Neste requisito, os sistemas devem permitir que somente eleitores registrados possam computar seus votos.

Em "Towards Secure E-Voting Using Ethereum Blockchain" (KOÇ et al, 2018), podemos notar na função de votar abaixo que existe a checagem `sender.hasRightToVote` (em negrito no código abaixo) que verifica se o eleitor se registrou anteriormente:

```
function vote(uint8 toProposal) public {
    Voter storage sender = voters[msg.sender];
    if (sender.isVoted || toProposal >= proposals.length &&
!sender.hasRightToVote)
        return;
    sender.isVoted = true;
    sender.vote = toProposal;
    proposals[toProposal].voteCount += 1;
}
```

O registro é feito através da seguinte função, que corretamente verifica se quem a executa é o presidente da eleição (`chairPerson`):

```
function giveRightToVote(address toVoter) public {
    if (msg.sender != chairPerson || voters[toVoter].isVoted){
        return;
    }
    else{
        voters[toVoter].hasRightToVote = true;
```

```
        }
    }
```

Em “Decentralized Voting Platform Based on Ethereum Blockchain” (KHOURY et al, 2018), a função de voto verifica se o eleitor foi registrado através da chamada da função `isEligble` (destacada em negrito):

```
    function VoteFor(bytes32 candidate) public payable {
        if ((verify[msg.sender] == true) || (completed == true)
    || (msg.value == 0) || !isEligble(msg.sender))
            revert();
        else {
            votes[candidate] += 1;
            totalNumVotes += 1;
            verify[msg.sender] = true;
        }
    }
```

A função `isEligble` por sua vez, verifica se o eleitor se encontra entre aqueles que finalizaram o registro:

```
    function isEligble(address _address) public view returns(bool){
        if (EligbleVotersAddresses[_address] == true)
            return true;
        else return false;
    }
```

Em “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021), dada a função de voto:

```
101  /// Registered User Participant Voting Some Candidate
102  function vote(string memory _electionId, address _candidateAddress, address _voterAddress) public {
103      require(elections[_electionId].electionIsActive, "Election Periode Has Ended!");
104      require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
105      require(elections[_electionId].electionParticipants[_voterAddress].voterWeight > 0, "You Are Not Registered!");
106      require(elections[_electionId].electionParticipants[_voterAddress].voterVoted == false, "You Already Voted!");
107      elections[_electionId].electionParticipants[_voterAddress].voterVoted = true;
108      elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate = _candidateAddress;
109      elections[_electionId].electionCandidates[_candidateAddress].candidateVoteCount
110          += elections[_electionId].electionParticipants[_voterAddress].voterWeight;
111  }
112
```

Podemos ver que existe uma checagem que verifica se o peso do voto do eleitor é maior que zero, e caso falhe, retorna uma mensagem de que o eleitor não está registrado:

```
    require(elections[_electionId].electionParticipants[_voterAddre
ss].voterWeight > 0, “You Are Not Registered!”)
```

Embora o processo de registro não esteja descrito no texto, pode-se assumir que durante esse processo, o peso do voto do eleitor recebe um valor maior que zero.

Em “A Smart Contract for Boardroom Voting with Maximum Voter Privacy” (McCORRY, SHAHANDASHTI e HAO, 2017), a função de votar, `submitVote`, pode ser vista na linha 941 do contrato `AnomymousVoting.sol`, e pode-se notar que existe uma checagem (em negrito) para ver se o eleitor está registrado:

```
     function submitVote(uint[4] params, uint[2] y, uint[2] a1,
uint[2] b1, uint[2] a2, uint[2] b2) inState(State.VOTE) returns
(bool) {

          // HARD DEADLINE
          if(block.timestamp > endVotingPhase) {
            return;
          }

          uint c = addressid[msg.sender];

          // Make sure the sender can vote, and hasn't already
voted.
          if(registered[msg.sender] && !votecast[msg.sender]) {
          …
```

Ou seja, todos os trabalhos atendem a este requisito.

5.1.6 Eleitores podem votar somente uma única vez

Neste requisito, os sistemas devem checar se, durante o voto, o eleitor ainda não votou, bem como alterar seu status de forma a não permitir um segundo voto.

Em “Towards Secure E-Voting Using Ethereum Blockchain” (KOÇ et al, 2018), podemos notar na função de votar abaixo que existe a checagem `sender.isVoted` (em negrito no código abaixo) que verifica se o eleitor se registrou anteriormente, e também a alteração de status (para bloquear novos votos, também destacada em negrito):

```
     function vote(uint8 toProposal) public {
         Voter storage sender = voters[msg.sender];
         if (sender.isVoted || toProposal >= proposals.length &&
!sender.hasRightToVote)
             return;
         sender.isVoted = true;
         sender.vote = toProposal;
         proposals[toProposal].voteCount += 1;
```

```
    }
```

Em “Decentralized Voting Platform Based on Ethereum Blockchain” (KHOURY et al, 2018), a função de voto verifica o voto anterior através do endereço do eleitor no *array* verify (destacada em negrito), e logo em seguida altera seu status (também destacado em negrito):

```
    function VoteFor(bytes32 candidate) public payable {
        if ((verify[msg.sender] == true) || (completed == true)
    || (msg.value == 0) || !isEligble(msg.sender))
            revert();
        else {
            votes[candidate] += 1;
            totalNumVotes += 1;
            verify[msg.sender] = true;
        }
    }
```

Em “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021), dada a função de voto:

```
101  /// Registered User Participant Voting Some Candidate
102  function vote(string memory _electionId, address _candidateAddress, address _voterAddress) public {
103      require(elections[_electionId].electionIsActive, "Election Periode Has Ended!");
104      require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
105      require(elections[_electionId].electionParticipants[_voterAddress].voterWeight > 0, "You Are Not Registered!");
106      require(elections[_electionId].electionParticipants[_voterAddress].voterVoted == false, "You Already Voted!");
107      elections[_electionId].electionParticipants[_voterAddress].voterVoted = true;
108      elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate = _candidateAddress;
109      elections[_electionId].electionCandidates[_candidateAddress].candidateVoteCount
110          += elections[_electionId].electionParticipants[_voterAddress].voterWeight;
111  }
112
```

Pode-se ver que existe uma checagem pelo endereço do eleitor em um array, e caso isso retorne `true`, a execução é interrompida:

```
    require(elections[_electionId].electionParticipants[_voterAddre
ss].voterVoted == false, “You Already Voted!”)
```

Logo após também é possível ver que esse valor é alterado para impedir novo voto do mesmo eleitor:

```
    elections[_electionId].electionParticipants[_voterAddress].
voterVoted = true;
```

Em “A Smart Contract for Boardroom Voting with Maximum Voter Privacy” (McCORRY, SHAHANDASHTI e HAO, 2017), a função de votar, `submitVote` (linha 941

do contrato `AnomymousVoting.sol)`, tem a checagem de verificação se o eleitor não votou (no *array* votecast, em negrito):

```
     function submitVote(uint[4] params, uint[2] y, uint[2] a1,
uint[2] b1, uint[2] a2, uint[2] b2) inState(State.VOTE) returns
(bool) {

          // HARD DEADLINE
          if(block.timestamp > endVotingPhase) {
            return;
          }

          uint c = addressid[msg.sender];

          // Make sure the sender can vote, and hasn't already
voted.
          if(registered[msg.sender] && !votecast[msg.sender]) {
          …
```

E na linha 917 da mesma função, a alteração de status do voto para impedir voto duplo:

```
          votecast[msg.sender] = true;
```

Novamente, todos os trabalhos atendem a este requisito.

5.1.7 Robustez

O requisito da robustez na rede Ethereum pode ser tratada sob dois aspectos diferentes: a escalabilidade (se a rede aguenta uma eleição de grandes proporções); e custo: será que os trabalhos levaram em conta o custo de implantação dos contratos e cada transação?

Até os dias atuais, nenhuma eleição de grandes proporções aconteceu na rede principal do Ethereum, mas quais dos trabalhos se preocuparam com este cenário?

Em “Towards Secure E-Voting Using Ethereum Blockchain” (KOÇ et al, 2018) os autores mostram a performance de apenas 3 eleitores na rede de testes Rinkeby (*Proof of Authenticity*), votando em 5 questões, onde o tempo de cada operação foi reportado em uma tabela:

	Criação do Contrato	**Eleitor 1 - Transação**	**Eleitor 3 - Transação**	**Eleitor 3 - Transação**
Voto 1	38s	33s	47s	49s
Voto 2	32s	32s	45s	45s
Voto 3	42s	49s	56s	56s
Voto 4	47s	36s	54s	54s
Voto 5	1m1s	32s	28s	28s

Tabela 1: Tempos de voto em "Towards Secure E-Voting Using Ethereum Blockchain"

Conforme os próprios autores, "nosso escopo é limitado para enquetes e eleições de pequena escala". Não há comentários sobre os custos envolvidos.

Em "Decentralized Voting Platform Based on Ethereum Blockchain" (KHOURY et al, 2018), que utilizou a rede de testes Ropsten (*Proof of Work*) não há descrição de quantos eleitores ou candidatos foram usados. Há uma citação sobre tempos de transação, com o registro dos eleitores tendo demorado entre 2 a 4 minutos, e o tempo de voto entre 40s e 2 minutos.

Também não há comentários sobre os custos envolvidos.

Em "Go-Ethereum for electronic voting system using clique as proof-of-authority" (CHRISTYONO, WIDJAJA e WICKSANA, 2021), os autores claramente não usaram a rede principal, mas sim uma rede de testes que fornecesse validação via *Proof of Authority*. Segundo eles, essa rede foi configurada com 3 mineradores e transações de 5 segundos (bastante abaixo do tempo médio da rede principal, que é em torno de 10 segundos).

Para os dados dos testes, os autores usaram dados reais da eleição para representante do conselho regional de Jelupang, em 2019, na Indonésia. Foram 26 candidatos e 15.725 eleitores.

O processo de criar uma conta nessa rede de testes para cada um dos eleitores levou em média 3 minutos, num total de 13.4 horas. Após isso, os autores afirmam que o processo de voto e confirmação para os 15.725 eleitores levou um total de 12,75 minutos (o que não parece bater com o cálculo de 15.725 eleitores levando 5 segundos por transação de voto e tendo apenas 3 mineradores, o que daria mais de 7 horas).

De qualquer forma, segundo os autores, numa projeção linear, a eleição na Província de Banten, que em 2019 teve 5,1 mi de eleitores, teria o tempo total de voto requerido para

processar as transações de todos os eleitores de 71 horas (muito mais que as usuais 5 horas naquele país).

Os autores não citam quais seriam os custos da eleição.

Em "A Smart Contract for Boardroom Voting with Maximum Voter Privacy" (McCORRY, SHAHANDASHTI e HAO, 2017) os autores dedicam um capítulo para o experiment de seu código na rede Ropsten (*Proof of Work*), que possui tempo de transação similar a rede principal. Segundo eles, foi preciso separar o algoritmo em dois contratos (um com a lógica de voto, e outro com a lógica de Prova de Zero Conhecimento Prévio), devido ao limite dos blocos da rede.

Foram simulados 40 eleitores votando em 2 opções ("sim" e "não"), com um total de 126 transações no total (entre implantação, registro, processos de voto, totalização).

Os tempos médios foram descritos em uma tabela:

Ação	**Tempo Médio (ms)**
Criação das Chaves	81
Registro das Chaves de Votos	142
Início da Eleição	277
Prova de Zero Conhecimento Prévio dos Votos	461
Votar	573
Totalização dos Votos	132

Tabela 2: Tempos médios de execução em "A Smart Contract for Boardroom Voting with Maximum Voter Privacy"

Os autores ainda calculam qual seria o número máximo de eleitores que seu algoritmo conseguiria suportar. Segundo eles, o valor é linear e progride de acordo com o número de eleitores, sendo limitado pelo valor máximo de *gas* (a unidade de custo do Ethereum) por transação.

Em 2016 os autores calcularam que, devido ao limite de 4.7 mi de *gas* por transação, o sistema atenderia um máximo de 60 eleitores (idealmente, 50, para se ter uma margem de segurança). Neste momento atual, o limite é de 30 mi, o que permitiria cerca de 383 eleitores.

Em termos de custos, o total gasto com 40 eleitores e 126 transações foi de 145.381.858 *gas*. Tirando os processos de administrativos (registro, geração de chaves, etc),

cada eleitor custou 3.323.642 *gas*. Em 2016, isso deu um valor de U$ 0,73 por eleitor e um total de custos pela eleição de U$ 31,98, já que a moeda do Ethereum, o Ether, custava U$ 11.

Nos dias atuais, o custo do Ether é de U$ 2.703,57, o que daria um custo total aproximado de U$ 7.858,00 pela mesma eleição com 40 eleitores.

5.1.8 Confiabilidade

No critério de Confiabilidade, TAŞ e TANRIÖVER (2020) propõe que o software para votação deva ser desenvolvido "sem nenhum código malicioso ou erros".

Por mais louvável que seja a intenção, é muito difícil produzir um software totalmente livre de erros, mas com um processo focado bastante em qualidade de software, é possível minimizar bastante a ocorrência. Ou seja, em confiabilidade, se espera que o sistema de voto esteja o tão livre de bugs quanto possível.

Ainda tomando como base a análise da urna brasileira feita por ARANHA et al (2014), um item muito importante deve ser incluído em qualquer análise de um sistema que tenha pretensões de executar uma eleição à nível nacional:

> *"Processo de desenvolvimento de software deficiente: processo de desenvolvimento ruim, que permite que vulnerabilidades acidentais ou maliciosas possam ser inseridas no software".*

Já que qualquer *smart contract* tem seu código fonte disponível na rede Ethereum, o mínimo que poderia se esperar de um sistema de *e-voting* no quesito de processo de desenvolvimento de software, seria que este código estivesse sob um controle de versão e ainda disponível em algum repositório de código fonte aberto a participações (*forks*, envio de erros, perguntas, wikis de usuários) como, por exemplo, o Github ou o Gitlab.

O que se assume dos trabalhos apresentados é que estejam, no mínimo, funcionais. Mas além disso, estariam estes trabalhos cientes das próprias deficiências? Estariam atentos a problemas comuns dos *smart contracts*?

Um exemplo comum que poderia atingir os sistemas de voto: todo *smart contract* pode ser instanciado por outro *smart contract*. Sem as devidas checagens, qualquer usuário da rede Ethereum poderia se registrar em uma das eleições.

Durante o voto: as funções verificam que apenas quem está votando é realmente quem está executando a função de voto?

Outro ponto que vários sistemas de voto em *blockchain* costumam esquecer: existe o conceito de eleição finalizada? Ou seja, após encerrar a eleição, o sistema bloqueia novos votos?

E ainda por último, será que os trabalhos apontam possíveis deficiências em suas implementações?

Em "Towards Secure E-Voting Using Ethereum Blockchain" (KOÇ et al, 2018), a função de registro corretamente verifica se quem a executa é o presidente da eleição (`chairPerson`), mas também não há checagem por estado da eleição ou limite de tempo:

```
function giveRightToVote(address toVoter) public {
    if (msg.sender != chairPerson || voters[toVoter].isVoted){
        return;
    }
    else{
        voters[toVoter].hasRightToVote = true;
    }
}
```

Já a função de voto, embora identifique corretamente o eleitor como o executor da função (`msg.sender`), não parece levar em conta se a eleição está em fase de votação ou algum limite de tempo:

```
function vote(uint8 toProposal) public {
    Voter storage sender = voters[msg.sender];
    if (sender.isVoted || toProposal >= proposals.length &&
!sender.hasRightToVote)
        return;
    sender.isVoted = true;
    sender.vote = toProposal;
    proposals[toProposal].voteCount += 1;
}
```

Embora os autores afirmem que seu "contrato tenha funções para ajustar o tempo de duração da eleição, como por exemplo: 120 minutos ou 3000 minutos", em nenhum código isso está presente.

Sobre suas próprias deficiências, os autores se limitam a citar a escabilidade desconhecida do Ethereum, e a falta de privacidade do modelo proposto.

Conforme já citado anteriormente, o trabalho não conta com um repositório e o código fonte está contido parcialmente no trabalho.

Em “Decentralized Voting Platform Based on Ethereum Blockchain” (KHOURY et al, 2018), a função de voto corretamente trata o eleitor como o executor da função (`msg.sender`). Mas tanto em sua função de voto quanto em sua função de registro, não há checagens de estado da eleição ou limites de tempo. Além disso, qualquer usuário poderia se registrar, já que não há a noção de um administrador da eleição. Abaixo a funções de voto:

```
function VoteFor(bytes32 candidate) public payable {
    if ((verify[msg.sender] == true) || (completed == true)
|| (msg.value == 0) || !isEligble(msg.sender))
        revert();
    else {
        votes[candidate] += 1;
        totalNumVotes += 1;
        verify[msg.sender] = true;
    }
}
```

O contrato que faz o registro pode ser visto no Apêndice B – Contrato Register.

Como deficiência, os autores não citam nenhuma especificamente, mas citam que o sistema poderia ter seu desenvolvimento levado adiante para acomodar eleições nacionais.

O trabalho não conta com um repositório, e o código fonte está incluído como apêndice.

Em “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021), dada a função de voto:

```
101  /// Registered User Participant Voting Some Candidate
102  function vote(string memory _electionId, address _candidateAddress, address _voterAddress) public {
103      require(elections[_electionId].electionIsActive, "Election Periode Has Ended!");
104      require(_voterAddress == msg.sender, "This Is Not Your Ballot!");
105      require(elections[_electionId].electionParticipants[_voterAddress].voterWeight > 0, "You Are Not Registered!");
106      require(elections[_electionId].electionParticipants[_voterAddress].voterVoted == false, "You Already Voted!");
107      elections[_electionId].electionParticipants[_voterAddress].voterVoted = true;
108      elections[_electionId].electionParticipants[_voterAddress].voterSelectedCandidate = _candidateAddress;
109      elections[_electionId].electionCandidates[_candidateAddress].candidateVoteCount
110          += elections[_electionId].electionParticipants[_voterAddress].voterWeight;
111  }
112
```

O sistema corretamente checa (na linha 104) se `_voterAddress` é igual ao `msg.Sender`. Pode-se ver que parece haver algum gerenciamento de estado simples, onde há uma checagem para ver se a eleição está ativa (`electionIsActive`, na linha 103), mas o trabalho não disponibiliza a função de registro dos eleitores ou ainda de que forma o estado da eleição é gerenciado.

Como deficiência, os autores apontam que se o sistema fosse usado para uma eleição maior (como na Província de Banten), o processamento dos votos levaria um tempo que eles não consideram adequado (71 horas).

O código fonte está incluído apenas parcialmente e apenas como uma figura.

Em "A Smart Contract for Boardroom Voting with Maximum Voter Privacy" (McCORRY, SHAHANDASHTI e HAO, 2017), ao analisar o contrato `AnonymousVoting.sol`, que está disponível de maneira integral no repositório do projeto (McCORRY e JOHNSON, 2017), pode-se ver que existe um gerenciamento de estado da eleição (`SETUP`, `SIGNUP`, `COMMITMENT`, `VOTE`, `FINISHED`). Também é estabelecido que o *owner* do contrato seja o gerenciador da eleição, e somente ele pode registrar os eleitores e iniciar/avançar/encerrar a eleição.

Dada a função de voto (parcialmente abaixo):

```
  function submitVote(uint[4] params, uint[2] y, uint[2] a1, uint[2]
b1, uint[2] a2, uint[2] b2) inState(State.VOTE) returns (bool) {

     // HARD DEADLINE
     if(block.timestamp > endVotingPhase) {
       return;
     }

     uint c = addressid[msg.sender];

     // Make sure the sender can vote, and hasn't already voted.
     if(registered[msg.sender] && !votecast[msg.sender]) {
```

É possível ver que a declaração da função já requer que ela só possa ser executada se o sistema estiver no estado `VOTE` (em negrito). Logo após, há a checagem por tempo (também em negrito). E o eleitor é sempre tratado como o executor da função (`msg.sender`) para evitar que outra pessoa possa votar por ele.

Os autores apontam várias deficiências e quais medidas foram tomadas para evitar os problemas. Entre as deficiências não resolvidas são apontados os seguintes problemas:

- O computador que o eleitor usa para votar pode estar comprometido;

- O eleitor pode tornar sua chave criptográfica privada pública e revelar seu voto;

- Se a fase de comprometimento (onde o voto do eleitor é salvo de forma cifrada, para evitar que ele mude o voto posteriormente) não for obrigatória, o último eleitor pode fazer a totalização antes de votar;

- O voto é obrigatório: se alguém deixar de votar, o sistema não consegue finalizar a eleição, porque as etapas criptográficas requerem todos os votos! Os autores sugerem que isso é evitado pelo sistema pelo fato dele cobrar um valor do eleitor (que depois será restituído se ele votar), e também sugerem estudos posteriores de novos modelos. Mas que não deixa de ser um grande ponto contra o modelo atual em eleições com milhares de eleitores, já que estatisticamente a chance de todos comparecerem é baixa, mesmo sob penalidade de multa;

- Os autores citam alguns possíveis problemas inerentes ao Ethereum: mineradores que possam, utilizando táticas como *selfish mining* ou *feather forking*, atrasar o processamento de blocos, ou seja, impedir que votos válidos tenham sua transação processada;

- O já citada falta de escalabilidade da solução: segundo dos autores, boa parte do processamento e uso de *gas* é devido à falta de recursos matemáticos (funções elípticas) usados nas provas de zero conhecimento prévio.

O projeto conta com um repositório Github de código fonte, e os autores costumam responder questionamentos, sugestões e críticas. Inclusive existe um tutorial em vídeo a ser seguido (embora possivelmente esteja desatualizado).

O código também inclui a interface *web* responsável por interagir com os usuários.

Ao analisar o repositório da solução, porém, é possível achar um defeito (Github, 2018) reportado por um usuário que diz ser impossível rodar o código na rede Ethereum atual, depois de atualizações do protocolo da rede feitas em 2018. Os autores dizem que, se o código for atualizado para usar as funções elípticas nativas do Ethereum (que foram adicionadas após a conclusão do trabalho e rodam com melhor desempenho e menor custo), é possível que ele volte a funcionar (os autores inclusive estão abertos a envios por voluntários).

5.1.9 Verificabilidade

Aqui um dos requisitos mais simples de ser analisado. Como os 3 primeiros trabalhos da lista, “Towards Secure E-Voting Using Ethereum Blockchain” (KOÇ et al, 2018), “Decentralized Voting Platform Based on Ethereum Blockchain” (KHOURY et al, 2018) e “Go-Ethereum for electronic voting system using clique as proof-of-authority” (CHRISTYONO, WIDJAJA e WICKSANA, 2021) não contemplam nenhum esquema de privacidade para proteger o voto do usuário, basta com que qualquer um em posse do hash do eleitor verifique o candidato escolhido pelo mesmo.

Já em "A Smart Contract for Boardroom Voting with Maximum Voter Privacy" (McCORRY, SHAHANDASHTI e HAO, 2017), como os valores são cifrados, os autores afirmam que basta o usuário usar a função de sua chave de decifração que ele pode verificar se o valor do seu voto foi armazenado corretamente.

5.2 Resultados

É possível resumir os critérios analisados em uma tabela, com os trabalhos validados nas colunas e os critérios nas linhas:

Trabalho / Critério	Towards Secure E-Voting Using Ethereum Blockchain	Decentralized Voting Platform Based on Ethereum Blockchain	Go-Ethereum for electronic voting system using clique as proof-of-authority	A Smart Contract for Boardroom Voting with Maximum Voter Privacy
Não produzir recibo	✓	✓	✓	✓
Não produzir resultados preliminares			? (*)	✓
Integridade de dados	✓	✓		✓
Privacidade/votação anônima				✓
Somente eleitores registrados podem votar	✓	✓	✓	✓
Eleitores podem votar somente uma única vez	✓	✓	✓	✓
Robustez				
Confiabilidade				✓ (**)
Verificabilidade	✓	✓	✓	✓

* Não há dados suficientes para analisar.

** Em seu estado atual, o código precisa ser atualizado para funcionar.

Tabela 3: Resultados

A quase totalidade dos critérios é atendida apenas em "A Smart Contract for Boardroom Voting with Maximum Voter Privacy" (McCORRY, SHAHANDASHTI e HAO, 2017), mas devido às mudanças no protocolo da rede Ethereum, em seu formato atual, o

código do sistema não consegue ser rodado por limitações da própria rede. Também é o único que conta com um repositório de código fonte colaborativo, como o Github, que já possui embutida a dinâmica de controle de versão, reportar erros/críticas/sugestões, delegar tarefas a usuários, suportar wikis, etc.

Também é o único a incluir, no repositório, o código de uma interface web para permitir a interação do eleitor (e administrador) com os *smart contracts*.

Notavelmente, nenhum dos modelos atendeu ao requisito Robustez, necessário para eleições de grandes proporções.

6 CONCLUSÃO

Nenhum dos modelos propostos atende a todos os requisitos propostos por TAŞ e TANRIÖVER (2020). Alguns não fornecem nem mesmo o código fonte completo, o que deveria incluir também uma interface que permita ao eleitor (e administrador) interagir com os *smart contracts* do Ethereum, já os *smart contracts* por padrão contém apenas a lógica e requerem uma interface adicional para interagir com os usuários.

Nos modelos que relatam uma interface, esta é web, o que traria o problema da confiabilidade do navegador em que ela está rodando: o dispositivo não pode conter nenhum software malicioso que possa intervir na eleição.

Isso sem considerar os inconvenientes dos sistemas precisarem estar conectados a Internet para realizar seu voto (possíveis ataques de *man in the middle*, DDoS, etc).

Em vista disso, estaria algum dos modelos propostos aptos a substituir a urna eletrônica brasileira?

A resposta é: da forma em que foram propostos não. O trabalho que mais se aproxima, "A Smart Contract for Boardroom Voting with Maximum Voter Privacy" (McCORRY, SHAHANDASHTI e HAO), que é o único a focar nos problemas de privacidade das atuais redes de *blockchain*, esbarra no problema de robustez, que segundo os autores, é penalizado pelas funções elípticas usados nas provas de zero conhecimento prévio, necessário para proteger a privacidade dos eleitores.

Alguns autores defendem que, inclusive, há um "trilema" da escalabilidade (WIKIPEDIA) inalcançável que as redes *blockchains* sofrem: elas não conseguiriam atender aos requisitos de descentralização, segurança e escalabilidade ao mesmo tempo (ou seja, um destes requisitos seria sempre sacrificado em prol dos outros dois).

Futuros trabalhos para sistemas de votos eletrônicos em uma *blockchain* voltada a execução de códigos (como o Ethereum) poderiam seguir dois caminhos: explorar a atualização do modelo de privacidade de votos proposto por McCORRY, SHAHANDASHTI e HAO, utilizando as (agora disponíveis) funções elípticas nativas do Ethereum, ou ainda focar em novas redes *blockchain* focadas em privacidade do usuário, como a Cardano.

REFERÊNCIAS

MOURA, Teogenes; GOMES, Alexandre. **Blockchain voting and its effects on election transparency and voter confidence**. Em: dg.o 17 - Proceedings of the 18th Annual International Conference on Digital Government Research, June 2017, 2017, Online. Association for Computing Machinery, 2017. p. 574–575. ISBN 9781450353175. Disponível em: https://doi.org/10.1145/3085228.3085263.

CHAUM, David L. **Computer Systems Established, Maintained and Trusted by Mutually Suspicious Groups**. 1979. 11f. Memorando No. UCB/ERL M79/10, College of Engineering, University of California, Berkeley, 1979.

NAKAMOTO, Satoshi. **Bitcoin: A Peer-to-Peer Electronic Cash System**. Bitcoin.org, 2008. Disponível em https://bitcoin.org/bitcoin.pdf. Acesso em: 19 de Abril de 2022.

ARANHA, Diego F. et al. **Software vulnerabilities in the Brazilian voting machine**. Em Design, Development, and Use of Secure Electronic Voting Systems, IGI Global, p. 149-175, março, 2014. Disponível em https://doi.org/10.4018/978-1-4666-5820-2.

TRIBUNAL SUPERIOR ELEITORAL. **Urna eletrônica 25 anos: lançado em 1996, equipamento é o protagonista da maior eleição informatizada do mundo**. Disponível em https://www.tse.jus.br/imprensa/noticias-tse/2021/Maio/urna-eletronica-25-anos-lancado-em-1996-equipamento-e-o-protagonista-da-maior-eleicao-informatizada-do-mundo. Acesso em: 19 de Abril de 2022.

HABER, Stuart; STORNETTA, Scott W. **How to Time-Stamp a Digital Document**. Advances in Cryptology, CRIPTO ’90, Springer-Verlag, LNCS 537, p 437-455, 1991.

BAYER, Dave; HABER, Stuart; STORNETTA, Scott W. Improving the Efficiency and Reliability of Digital Time-Stamping. Em: Capocelli, R., De Santis, A., Vaccaro, U. (eds) Sequences II. Springer, New York, NY, 1993. Disponível em https://doi.org/10.1007/978-1-4613-9323-8_24.

DAI, Wei. **B-Money**. Disponível em http://www.weidai.com/bmoney.txt. Acessado em 20 de Abril de 2022.

CONSENSYS. **What are the Benefits of Blockchain in Real State?** Disponível em https://consensys.net/blockchain-use-cases/real-estate/. Acessado em 20 de Abril de 2022.

BUTERIN, Vitalik. **Ethereum Whitepaper**. Disponível em https://ethereum.org/en/whitepaper. Acessado em 21 de Abril de 2022.

WIKIPEDIA. **Electronic Voting**. Disponível em https://en.wikipedia.org/wiki/Electronic_voting. Acessado em 21 de Abril de 2022.

BLUM, Manuel; FELDMAN, Paul; MICALI Silvio. **Non-Interactive Zero-Knowledge and Its Applications**. Em Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC '88). Association for Computing Machinery, New York, NY, USA, p. 103–112. Disponível em https://doi.org/10.1145/62212.62222.

KOÇ, Ali K. et all. **Towards Secure E-Voting Using Ethereum Blockchain**. Em 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 2018, p. 1-7. Disponível em https://doi.org/10.1109/ISDFS.2018.8355340.

KHOURY, Davi et all. **Decentralized Voting Platform Based on Ethereum Blockchain** 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), 2018, p. 1-6. Disponível em https://doi.org/10.1109/IMCET.2018.8603050.

FAYEMI, Oluwatosin J.; THOMPSON, Aderonke FB.; AYENI, Olaniyi A. **Design and Implementation of Electronic Voting Using KECCAK256 Algorithm on Ethereum Network**. Em International Conference on Innovative Computing and Communications, 2021. Advances in Intelligent Systems and Computing, vol 1394, p 431-453. Springer, Singapore. Disponível em https://doi.org/10.1007/978-981-16-3071-2.

CHRISTYONO, Basilius B. A.;WIDJAJA, Moeljono; WICKSANA, Aria. **Go-Ethereum for electronic voting system using clique as proof-of-authority**. Em TELKOMNIKA Telecommunication, Computing, Electronics and Control, Vol 19, No. 5. Universitas Ahmad Dahlan, 4th Campus, Yogyakarta, Indonesia. ISSN: 1693-6930, e-ISSN: 2302-9293. Disponível em http://journal.uad.ac.id/index.php/TELKOMNIKA/article/view/20415.

McCORRY, Patrick; SHAHANDASHTI, Siamak F; HAO, Feng. **A smart contract for boardroom voting with maximum voter privacy**. Em: 21st International Conference on Financial Cryptography and Data Security (FC17), Sliema, Malta, 3-7 de Abril de 2017. Financial Cryptography and Data Security FC 2017, 10322, p. 357-375. ISBN: 9783319709710. Disponível em https://doi.org/10.1007/978-3-319-70972-7_20.

TECHCRUNCH.com. **Sierra Leone just ran the first blockchain-based election**. Disponível em https://techcrunch.com/2018/03/14/sierra-leone-just-ran-the-first-blockchain-based-election/. Acessado em 22 de Abril de 2022.

GOVERNMENT TECHNOLOGY.com. **Utah County Makes History With Presidential Blockchain Vote**. Disponível em https://www.govtech.com/products/utah-county-makes-history-with-presidential-blockchain-vote.html. Acessado em 22 de Abril de 2022.

TAŞ, Ruhi; TANRIÖVER, Ömer Ö. **A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting**. Em Symmetry 12, no. 8: 1328, 2020. Disponível em https://doi.org/10.3390/sym12081328.

McCORRY, Patrick; JOHNSON, John. **Open Vote Network**. Disponível em https://github.com/stonecoldpat/anonymousvoting. Criado em 23 de Agosto de 2017. Visitado em 22 de Abril de 2022.

GITHUB. **Gas consumption is too high to create AnonymousVoting contract!.** Publicado em 23 de Julho de 2018. Disponível em https://github.com/stonecoldpat/anonymousvoting/issues/22. Visitado em 4 de Maio de 2022.

WIKIPEDIA. Trilema da Escalabilidade. Disponível em https://pt.wikipedia.org/wiki/Trilema_da_escalabilidade. Atualizado em 21 de dezembro de 2018. Visitado em 4 de Maio de 2022.

Apêndice A – Função votar do contrato AnonymousVoting.sol

```
  function submitVote(uint[4] params, uint[2] y, uint[2] a1, uint[2]
b1, uint[2] a2, uint[2] b2) inState(State.VOTE) returns (bool) {

     // HARD DEADLINE
     if(block.timestamp > endVotingPhase) {
       return;
     }

     uint c = addressid[msg.sender];

     // Make sure the sender can vote, and hasn't already voted.
     if(registered[msg.sender] && !votecast[msg.sender]) {

       // OPTIONAL Phase: Voters need to commit to their vote in
advance.
       // Time to verify if this vote matches the voter's previous
commitment.
       if(commitmentphase) {

         // Voter has previously committed to the entire zero
knowledge proof...
         bytes32 h = sha3(msg.sender, params,
voters[c].registeredkey, voters[c].reconstructedkey, y, a1, b1, a2,
b2);

         // No point verifying the ZKP if it doesn't match the
voter's commitment.
         if(voters[c].commitment != h) {
           return false;
         }
       }

       // Verify the ZKP for the vote being cast
       if(verify1outof2ZKP(params, y, a1, b1, a2, b2)) {
         voters[c].vote[0] = y[0];
         voters[c].vote[1] = y[1];

         votecast[msg.sender] = true;

         totalvoted += 1;

         // Refund the sender their ether..
         // Voter has finished their part of the protocol...
         uint refund = refunds[msg.sender];
         refunds[msg.sender] = 0;

         // We can still fail... Safety first.
         // If failed... voter can call withdrawRefund()
         // to collect their money once the election has finished.
         if (!msg.sender.send(refund)) {
            refunds[msg.sender] = refund;
         }
```

```
          return true;
        }
      }

      // Either vote has already been cast, or ZKP verification
failed.
      return false;
  }
```

Apêndice B – Contrato Register

```
pragma solidity 0.4.24;
import "browser/OraclizeAPI1.sol";
import "github.com/willitscale/solidity-util/lib/Strings.sol";
contract Register is usingOraclize
{
      using Strings for string;
      mapping(address => bool) public EligbleVotersAddresses;
      mapping(string => bool) RegisteredPhoneNumbers;
      mapping(bytes32 => string) OraclizeIDtoResult;
      mapping(address => bytes32) GeneratePINOraclizeID;
      mapping(address => string) AddressesToVerify;

      function isEligble(address _address) public view returns(bool)
{
        if (EligbleVotersAddresses[_address] == true)
        return true;
        else return false;
      }

      function verify(string code) public {
        bytes32 OracleID = GeneratePINOraclizeID[msg.sender];
        if (OraclizeIDtoResult[OracleID].compareTo(code)) {
          EligbleVotersAddresses[msg.sender] = true;
          RegisteredPhoneNumbers[AddressesToVerify[msg.sender]] =
true;
        delete AddressesToVerify[msg.sender];
        delete OraclizeIDtoResult[OracleID];
        delete GeneratePINOraclizeID[msg.sender];
        }
      }

      function SendSMS(string phonenumber) public payable {
bytes32 OracleID;
        if (RegisteredPhoneNumbers[phonenumber] == true ||
phonenumber.compareTo(AddressesToVerify[msg.sender])
|| GeneratePINOraclizeID[msg.sender] == 0)
          revert();
        bytes32 ID = GeneratePINOraclizeID[msg.sender];
        string memory jsonStart = '{"To" : "';
        string memory jsonMiddle = '","From" : "+19103708518" ,
"Body": "';
        string memory jsonEnd = '"}';
        OracleID = oraclize_query('URL',
'json(https://API:PASS@api.twilio.com..../
Messages.json).status',
strConcat(jsonStart, phonenumber, jsonMiddle,
OraclizeIDtoResult[ID], jsonEnd));
        AddressesToVerify[msg.sender] = phonenumber;
      }

      function generatePIN() public payable {
        bytes32 OracleID;
        if (EligbleVotersAddresses[msg.sender] == true)
```

```
          revert();
        if (!AddressesToVerify[msg.sender].compareTo("0
x0000000000000000000000000000000000000000"))
          delete AddressesToVerify[msg.sender];
        OracleID = oraclize_query("WolframAlpha", "2 digit integer
random");
        GeneratePINOraclizeID[msg.sender] = OracleID;
      }

      function __callback(bytes32 _OraclizeID, string _result) public
{
        if (_result.compareTo("Queued") == false)
          OraclizeIDtoResult[_OraclizeID] = _result;
      }
}
```