UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Uma proposta de Arquitetura Distribuída para E-Voting

Trabalho de Conclusão de Curso

João Marcos Martins Soares

São Cristóvão – Sergipe

2021

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

João Marcos Martins Soares

Uma proposta de Arquitetura Distribuída para E-Voting

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Rafael Oliveira Vasconcelos

São Cristóvão – Sergipe

2021

Agradecimentos

Agradeço aos meus pais, Mirabel e João Nylson, aos meus irmãos João Guilherme e João Victor e também à Julia por me apoiarem, cada um ao seu jeito, durante minha graduação e durante o desenvolvimento deste trabalho.

Também agradeço aos professores, técnicos, demais funcionários do DCOMP. Agradeço principalmente ao meu orientador Rafael, pela enorme paciência e boa vontade. Em especial também agradeço à Giovanny, Breno, Kalil e Estombelo, que particulamente me marcaram e me inspiraram ao longo da graduação.

Resumo

Sistemas de e-voting apresentam vários desafios em relação à segurança, sigilo do voto e transparência. O objetivo deste trabalho é explorar os requisitos necessários para uma uma eleição eletrônica e analisar as vantagens e desvantagens que uma arquitetura distribuída pode agregar ao tema. O trabalho conclui que, em uma arquitetura distribuída, o mecanismo de consenso da rede e o armazenamento dos votos são pontos cruciais no projeto do sistema, e que invenções tecnológicas ainda são necessárias para atender todos os requisitos.

Palavras-chave: e-voting. eleição. sistemas distribuídos. consenso. criptografia.

Abstract

E-voting system presents a series of challenges regarding security, voter privacy and transparency. This paper explores the requirements of an electronic election and discuss the benefits and drawbacks of a distributed architecture. The paper concludes that, in the context of a distributed architecture, consensus mechanism and vote storage are critical aspects in system design, and technological advancements are still necessary in order to meet all requirements.

Keywords: e-voting. election. distributed systems. consensus. cryptography.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Propostas de sistemas eleitorais eletrônicos (e-voting) buscam diminuir os custos de uma eleição, mantendo requisitos de integridade, segurança e privacidade (HJáLMARSSON et al., 2018). Embora os benefícios de eficiência e custo em processos informatizados sejam claros em relação a processos analógicos, os cuidados de segurança podem se tornar mais complexos (STALLINGS, 2017).

A literatura aponta segurança como o desafio principal em e-voting. Trabalhos apontam os riscos teóricos do voto eletrônico em comparação com os riscos do sistema tradicional (LAUER, 2004; GRITZALIS, 2002). Outros trabalhos apontam falhas em implementações de sistemas de voto eletrônicos (SPRINGALL et al., 2014; ARANHA; GRAAF, 2018).

Nesse sentido, o advento da computação em nuvem e arquiteturas descentralizadas surgem como uma tentativa de lidar com os desafios de segurança impostos, principalmente relacionados a confiança no sistema. Com uma arquitetura descentralizada, as partes envolvidadas precisam entrar em um acordo sobre os valores que serão comitados. O compartilhamento de responsabilidades pode melhorar a confiança no sistema. Para isso, será apresentado como sistemas distribuídos em computação solucionam conflitos e como isso pode ser aproveitado para uma arquitetura descentralizada. Além disso, com o surgimento da tecnologia *blockchain*, trabalhos foram publicados sobre o uso dessa tecnologia no contexto de eleições eletrônicas (HJáLMARSSON et al., 2018; KSHETRI; VOAS, 2018).

A vantagem da arquitetura distribuída é espalhar a responsabilidade do sistema entre várias entidades, adicionando redundância. Este trabalho busca apresentar os requisitos e desafios que uma arquitetura distribuída de eleição deve possuir. Inicialmente foi planejado que a arquitetura proposta usaria blockchain como plataforma. Porém, percebeu-se que determidados requisitos de sistemas eleitorais fazem com que a aplicação pura de blockchain não seja ideal. Blockchain por sí só não provê mecanismos de autenticação, algo necessário para e-voting. Se a rede requer que os participantes sejam identificados, uma entidade central é necessária. Além disso, parte do

sigilo na blockchain (mais especificamente na rede Bitcoin) provém do fato de que os usuários não são autenticados e, por isso, podem assumir inúmeras identidades. Blockchain também não resolve o problema do armazenamento secreto dos votos (que será apresentado na seção 2.1.9 e discutido no início do capítulo 3). Ou seja, não há como garantir que o voto guardado na *chain* só ficará disponível com o fim da eleição. Assim, o benefício da blockchain fica restrito ao armazenamento distribuído e imutabilidade. Para o armazenamento distribuído, um banco de dados distribuído tradicional pode resolver. Um resultado similar ao da imutabilidade pode ser alcançado se o sistema entrar em consenso sobre as cifras dos votos antes da divulgação (Ver a seção 3.3.2).

1.1 Objetivos

O objetivo deste trabalho é apresentar uma proposta de arquitetura descentralizada e distribuída para eleições eletrônicas de pequeno porte, em que o número de responsabilidades de uma entidade central deve ser o menor possível. Para os fins desta pesquisa "eleições eletrônicas de pequeno porte"serão entendidas como as eleições internas de instituições públicas e privadas, organizações estudantis e sindicatos.

De forma a alcançar o objetivo geral, os seguintes objetivos específicos foram traçados:

a) Apresentar os requisitos de uma eleição eletrônica segura;

b) Propor um modelo distribuído de eleição baseado nos requisitos levantados;

c) Analisar os principais desafios impostos por esse modelo;

d) Analisar suas vantagens, desvantagens e viabilidade.

1.2 Metodologia

Para o desenvolvimento deste trabalho, foi realizada uma pesquisa exploratória sobre o e-voting em artigos da literatura. Após isso, foram estudados conceitos sobre sistemas distribuídos e criptogrfia. Por fim, foi proposta uma arquitetura baseada nos temas estudados.

1.3 Revisão Bibliográfica

Para o presente estudo, foram utilizados como base artigos sobre voto eletrônico. Gritzalis (2002) aponta um conjunto de princípios que o desenvolvimento de sistemas de eleição deve cumprir, baseado em princícios constitucionais de países democráticos. Além disso, o autor discute se e-voting deve ser usado como meio complementar ao processo tradicional ou de forma independente.

Lauer (2004) enumera os possíveis riscos do voto eletronico, tanto o voto pela Internet quanto o uso de tecnologia como suporte à eleição tradiconal. São analisados os riscos em níveis físicos e de rede, além de serem feitas recomendações sobre como mitigar esses riscos.

O trabalho de Springall et al. (2014) analisa a segurança do sistema de eleição online da Estônia, utilizando como base análise de código, observações presenciais e testes de invasão em réplicas do sistema. Os resultados mostram que existem falhas de segurança na arquitetura e que o desenho de sistemas eletrônicos de votação é díficil, pois deve garantir confiança no resultado ao mesmo tempo que é preciso manter o sigilo do voto.

Um trabalho similar é feito por Aranha e Graaf (2018), porém no contexto das eleições brasileiras. Os autores apresentam o processo eleitoral no Brasil e discutem pontos de falha envolvendo a urna eletrônica. É discutido que um dos pontos principais de falha está no processo de validação do código-fonte, pois não é garantido que o código fonte que está contido na urna é o mesmo auditado pelas partes interessadas. Além disso, o próprio processo de auditoria do código fonte é limitado.

O uso da computação em nuvem para eleições eletônicas é explorado no trabalho de Zissis e Lekkas (2011). Ainda neste trabalho, o uso de processos eletrônicos em governos é revisado, além de listar um conjunto de ameaças, como por exemplo código malicioso no cliente interceptação de mensagens, e suas respectivas medidas de segurança.

No contexto de computação em nuvem para eleições, alguns trabalhos estudam o uso de blockchain. Kshetri e Voas (2018) apresenta blockchain como uma possível solução para e-voting. Os autores argumentam que os eleitores podem enviar votos secreto a partir do próprio smarphone e que o sistema de blockchain proveria autenticação. Os resultados não poderiam ser modificados devido a propriedade de imutabilidade da blockchain e, portanto, os votos estariam seguros. O artigo é introdutório e não apresenta com clareza como a tecnologia pode ser usada, não traça paralelos com os requisitos de uma eleição e nem discutem posssíveis falhas. Os autores citam exemplos de eventos eleitorais que utilizaram blockchain na Russia, Coreia do Sul e Serra Leoa, mas as fontes não indicam artigos publicamente disponíveis que descrevam o processo eleitoral de cada uma delas.

Hjálmarsson et al. (2018) propõe um sistema de eleição utilizando uma rede blockchain permissionada, ou seja, uma rede em que somente usuários autenticados podem usar. Apesar de destacar que um dos requisitos da solução é a intraciabilidade, o trabalho não explicita como o voto é armazenado na rede, além de deixar claro que o eleitor pode futuramente verificar seu próprio voto com um ID gerado.

Hardwick et al. (2018) também explora o uso de blockchain para o armazenamento de votos em um esquema de e-voting. Para os autores, as características de anomicidade, transparência e segurança da blockchain se encaixam nos requisitos de uma eleição. Nesse esquema, apesar do objetivo ser criar uma rede mais distribuída possível, existe uma autoridade

central responsável por administrar as identidades dos eleitores. Além disso, para minimizar os efeitos da coercibilidade (O princípio da incorcibilidade será discutido na seção 2.1.3) o autor propõe que o voto fique atrelado com o eleitor até o final do eleição e que esse voto possa ser modificado livremente. Assim, se o eleitor for coagido no momento da emissão do voto, ele será capaz de alterá-lo futuramente num momento mais propício.

2

Fundamentação Teórica

Neste capítulo, serão apresentados os assuntos que servem de base para o trabalho. A seção 2.1 apresenta os princípios necessários para o desenvolvimento de um sistema eleitoral eletrônico seguro. A seções 2.2 e 2.3 apresentam o conceito de sistemas distribuídos e algoritmos de consenso. A seção 2.4 mostra o que são funções moderamente difíceis e suas aplicações. As seções 2.5 e 2.6 introduzem conceitos de criptografia.

2.1 Requisitos para uma Eleição Eletrônica Segura

O primeiro passo para a construção da arquitetura de um sistema de eleição seguro é definir os requisitos que ele seja considerado seguro e efetivo. Vários trabalhos na literatura (ZISSIS; LEKKAS, 2011; SAMPIGETHAYA; POOVENDRAN, 2006; HARDWICK et al., 2018) utilizam o trabalho de Gritzalis (2002) como referência para o mapeamento de requisitos.

Gritzalis (2002) define uma série de princípios de design que devem ser seguidos no desenvolvimento de sistemas de e-voting. Esses princípos são derivados de princípios constitucionais de países democráticos. Esses princípios ora se completam, ora se contradizem. Portanto, é impossível desenhar um sistema (seja tradicional ou eletrônico) que atenda completamente todos eles. Os princípios são listados a seguir.

2.1.1 Princípio do Isomorfismo ao Processo Tradicional

O processo eleitoral eletrônico deve garantir as características do processo eleitoral tradicional de sufrágio universal. Para isso, as seguintes premissas são definidas:

a) Todo eleitor tem o direito de participar do processo eleitoral;

b) O que define o indivíduo como eleitor deve ser encontrado e controlado em lei;

c) A tecnologia para lançar o voto deve ser acessível para todos os eleitores;

d) E-voting deve ser considerado um meio alternativo de voto;

e) O princípio democrático (ou seja, todo eleitor tem o direito de participar do processo eleitoral) leva à necessidade de uma infraestrutura pública adequada (pontos de Internet gratuita, etc).

Essas definições foram feitas para eleições nacionais. Para processos eleitorais mais restritos, como, por exemplo, os de uma universidade, a lei citada no item *b* das premissas acima pode ser substituida por um regimento interno. O importante é que as regras que definem quem pode votar existam. Consequentemente, no item *e*, o importante é garantir que os eleitores possuam acesso à infraestrutura necessária para exercer sua vontade política. Também é através desse princípio que se faz necessário ao sistema de votação garantir que o eleitor esteja autenticado para votar.

2.1.2 Princípio da Eligibilidade ao Voto

É necessário que os eleitores possam se registrar e se autenticar para emitir o voto. Esse princípio não entra em conflito com o princípio anterior, pois, para ser considerado um eleitor, é preciso que o indivíduo atenda aos critérios postos nas normas que definem quem pode votar naquelas eleições, além de impedir que um eleitor possa votar mais de uma vez.

2.1.3 Princípio da Incoercibilidade

Deve ser garantido que o voto não pode ser comprado e que o eleitor não pode ser extorquido. Esse princípio só pode ser alcançado se a pessoa eleitora não conseguir provar que ela votou em determinada proposta. Além dos requisitos técnicos para isso, a estrutura pública de votação (cabine eleitoral, por exemplo) deve garantir que a pessoa não possa ser coagida fora do sistema.

2.1.4 Princípio da Liberdade de Decisão

A liberdade de decisão do eleitor pode ser violada se alguma propaganda de partido ou candidato é veiculada durante a emissão do voto. No processo eleitoral tradicional, não é permitido veicular propaganda no local de votação. De maneira análoga, em um processo de votação eletrônico, não deve ser permitida a veiculação de propaganda eleitoral no *site* da votação ou, de um modo geral, na interface do sistema.

2.1.5 Princípio da Opção do Voto Inválido

O eleitor deve ter a liberdade de preferência preservada. Isso significa que, se nenhuma das opções disponíveis agradá-lo, ele deve poder emitir um voto inválido (nulo).

2.1.6 Princípio da Igualdade Entre Candidatos

A interface do sistema não deve favorecer ou prejudicar candidatos. Esse princípio é similar ao princípio da liberdade decisão, porém com foco nos candidados. O posicionamento das cédulas de voto em uma eleição tradicional e a exibição dos candidatos na tela de um computador devem prover igualdade entre os candidatos.

Além disso, todos os candidatos devem ter igual acesso à transparência da eleição. Ou seja, todos eles devem ter acesso às mesmas ferramentas e informações para verificar e auditar o funcionamento correto do processo eleitoral.

2.1.7 Princípio da Igualdade Entre Eleitores (Um Eleitor, Um Voto)

O princípio da igualdade entre os eleitores estabelece que todos os votos devem possuir o mesmo valor. Dessa premissa, são retiradas três informações. A primeira é que não deve existir um sistema de pesos em que o voto de determinado indivíduo vale mais que de outro. A segunda é que um eleitor pode emitir um, e somente um, voto.

Por fim, Phillips e Spakovsky (2001) discutem que, no contexto de uma votação eletrônica, o acesso e a familiaridade com a tecnologia podem fazer com que determinado grupo de eleitores tenha vantagem ao emitir o voto. Também por esse motivo, o acesso as ferramentas para emitir o voto devem ser universais, como discutido no princípio do isomorfismo. Por outro lado, para pessoas que por qualquer motivo físico (comorbidade, distancia geográfica) não possam comparecer a um local de votação tradicional, o sistema de e-voting pode ajudar a tornar verdadeiro o princípio da igualdade entre os eleitores.

2.1.8 Princípio do Sigilo

Esse princípio está diretamente ligado ao princípio da incoercibilidade e da liberdade de decisão. Sigilo é requisito necessário para garantir a decisão política livre. Uma vez lançado, o voto deve ser irreversível e nem o próprio eleitor deve ser capaz de recuperar sua decisão.

Em sistemas eleitorais tradicionais, o sigilo é garantido por barreiras físicas, entretanto, Gritzalis (2002) entende que em sistemas de e-voting o sigilo pode ser comprometido, e que os seguintes requisitos devem ser explicitamente atendidos:

a) O sigilo do voto deve ser garantido no lançamento, no transporte (nesse caso, por rede), na recepção e contagem;

b) Nenhum dos atores envolvidos (organizadores, candidatos e eleitores) deve ser capaz de fazer a conexão entre um eleitor e um voto;

c) Deve existir uma separação bem definida entre os processos de registro e autenticação e os processos de lançamento e coleta de votos;

d) Nenhum eleitor deve ser capaz de provar que votou em determinado candidato.

É pertinente ressaltar que em qualquer sistema votação remoto não existe garantia de que o indivíduo não será coagido no momento do lançamento do voto, para isso, ao desenhar sistemas de e-voting, medidas externas de proteção devem ser discutidas, como cabines de votação (GRITZALIS, 2002).

2.1.9 Princípio do Voto Não Monitorado

Durante o processo eleitoral, o voto não pode ser monitorado por terceiros. Nas eleições tradicionais, esse princípio é imposto com medidas que garantem que a urna só será aberta no momento da contagem de votos. Em um contexto de votação eletrônica, é preciso garantir que os votos computados só possam ser lidos após determinado momento, que indique o fim da recepção dos votos. Caso isso não ocorra, a entidade que armazena os votos poderá realizar monitoração.

2.1.10 Princípio da Transparência

O processo eleitoral de e-voting deve ser o mais transparente possível e a *fé no sistema* a menor possível. Ou seja, numa situação ideal, todos os atores envolvidos (organizadores, candidatos e eleitores) devem ser capazes de entender como a eleição ocorre. Segundo Gritzalis (2002), a *fé no sistema* é um fator inerente à votação eletrônica, dado que um cidadão comum não possui conhecimento sobre como sistemas computacionais funcionam.

Por isso, é importante que o sistema eleitoral eletrônico seja transparente e que os processos envolvidos sejam explicados, mesmo que em linguagem alto nível. Essa caraterística deve acompanhar o sistema em si.

2.1.11 Princípio da Verificabilidade e Prestação de Contas

Verificabilidade é a capacidade de eleitores, candidatos, organizadores e observadores independentes de verificarem que o processo eleitoral ocorre da maneira correta. A prestação de contas está relacionada à capacidade do sistema de produzir *logs* e monitoramento de todas as operações da eleição, de modo que qualquer indíviduo possa buscar por inconsistências.

Porém, a possibilidade do eleitor de verificar que seu voto foi computado e contado entra em completa contradição com o princípio do sigilo e da incoercibilidade. De um modo geral, os sistemas eleitorais (tanto tradicionais, quanto eletrônicos) realizam um *trade off* e abrem mão de parte da verificabilidade e prestação de contas para manter o sigilo do voto.

2.1.12 Princípio da Simplicidade

O princípio da simplicidade declara que o sistema deve ser o mais simples possível. A simplicidade, então, atrelada ao princípio da transparência, busca garantir que o sistema seja *user-friendly*, ou seja, simples, acessível e compreensível para o usuário.

2.1.13 Princípio da Confiabilidade e Segurança

Esse princípio permeia vários outros princípios supracitados. A confiabilidade significa que o resultado da eleição é realmente o reflexo da vontade do eleitorado, portanto, o sistema deve garantir que o resultado da eleição está de acordo com os votos lançados e que estes não foram modificados e são fiéis ao que os eleitores desejam. Não deve ser permitida a exclusão de nenhum voto emitido, e os votos nulos devem ser contabilizados, em conformidade com o princípio da opção do voto inválido.

A segurança refere-se à garantia de sigilo, integridade e disponibilidade do sistema. O processo de registro e autenticação dos atores deve ser feito de maneira segura e o sistema deve estar disponível e protegido contra ataques de negação de serviço, intencionais ou não, uma vez que a indisponibilidade do sistema fere o direito do eleitor de lançar seu voto.

De certo modo, A segurança contrapõe o princípio da simplicidade, já que processos de segurança inevitavelmente acabam aumentando a complexidade do sistema.

Baseado no princípio da confiabilidade e segurança, e na verificabilidade e prestação de contas, Gritzalis (2002) propõe os seguintes requisitos:

a) Devem existir processos de certificação de *hardware* e *software*;

b) Toda a infraestrutura, assim como as funcionalidades do sistema devem ser verificáveis (por exemplo, *open-source*);

c) Todas as operações devem ser monitoradas e gerar *logs* de sistema, exceto aquelas que possam quebrar o sigilo do voto;

d) A infraestrutura deve ser aberta para inspeção de agentes autorizados;

e) Deve ser garantido aos eleitores e candidatos que não houve más práticas durante o processo.

2.2 Consenso em Sistemas Distribuídos

Um sistema distribuído é uma coleção de computadores independentes que cooperam para resolver um problema que não pode ser resolvido individualmente (KSHEMKALYANI; SINGHAL, 2008). Mais precisamente, as entidades de um sistema distribuído possuem as seguintes características:

a) **Relógio físicos independentes**: Os relógios da rede não estão necessariamente sincronizados, isso torna-se um obstáculo para gerenciar a ordem das mensagens trocadas. Esse problema foi a motivação para o trabalho de Lamport (1978) que deu origem à teoria do tempo lógico.

b) **Memórias independentes**: As entidades não compartilham mémoria, ou seja, toda a comunicação depende de mensagens.

c) **Autonomia e heterogeneidade**: Os processos possuem velocidades diferentes e podem rodar em sistemas operacionais diferentes, com implementações diferentes.

Como o meio de comunicação é o aspecto mais importante da rede, é preciso classificar os erros que podem ocorrer durante a transmissão de mensagens. O pior tipo de erro nesse contexto é a *falha bizantina*, definida por Lamport (1983). Nessa falha, os nós podem apresentar qualquer erro arbitrário na troca de mensagens, inclusive alteração proposital do conteúdo. Numa variação desse problema, a *falha bizantina com autenticação*, os mesmo erros podem ocorrer, mas as mensagens podem ser verificadas através de assinaturas.

Um dos grandes problemas de sistemas distribuídos é o consenso entre os nós. Como uma rede de nós concorda sobre determinado valor? No contexto de falhas bizantinas, o algoritmo proposto mais conhecido é o *Practical Byzantine Fault Tolerance*, ou PBFT, proposto por Castro, Liskov et al. (1999). Nesse algoritmo, é possível garantir consenso em até $\frac{n-3}{3}$ nós maliciosos, sendo n o número total de nós. Outro algoritmo de consenso proposto depois é o *Proof-of-Work*, que será trabalhado em mais detalhes na seção abaixo.

2.3 Blockchain e Proof-of-Work

Blockchain é uma estrutura de dados *append-only*, ou seja, a única modificação permitida é a adição de mais dados. Os dados são inseridos em blocos, de maneira que cada bloco possui um *hash* do bloco anterior. É dessa estrutura que surge o nome *blockchain*: *block* = bloco e *chain* = corrente (NAKAMOTO, 2008).

Como a rede não possui uma entidade central, os nós precisam decidir em conjunto qual deles irá adicionar o próximo bloco na corrente. A solução mais intuitiva é um sorteio. Porém essa forma de seleção randômica é suscetível a ataques, além necessitar que algum nó assuma um papel centralizador (ZHENG et al., 2017).

É utilizado então o algoritmo *Proof-of-work* (DWORK; NAOR, 1993)[1], onde os nós precisam provar que realizaram determinado trabalho para gerar um novo bloco e adicioná-lo à rede. Para esse algoritmo, o trabalho também pode ser interpretado como gasto de tempo, expresso em ciclos de CPU. Em trabalhos posteriores, a ideia do algoritmo é extendida de modo que o nó possa gastar outros recursos, como espaço ou moedas digitais (DZIEMBOWSKI et al., 2015) (GAŽI; KIAYIAS; ZINDROS, 2019).

Para o bitcoin, é requisito da rede que os *hashs* dos blocos devem obrigatoriamente começar com uma determinada quantidade zeros. Essa quantidade de zeros no começo do *hash* é chamada de objetivo. Para isso, além do histórico de transações, os blocos precisam ter um elemento chamado *nonce*. O trabalho que o nó deve exercer é achar um *nonce* que, somado com

[1] Na literatura, o termo *Proof-of-Work* é usado em dois contextos. No primeiro, como uma forma de provar que uma máquina exerceu trabalho, criado por Dwork e Naor (1993). No segundo, adaptado por Nakamoto (2008) como um algoritmo de consenso.

os outros dados do bloco, gere um *hash* com a quantidade pré-definida de zeros no começo. O trabalho a ser realizado é exponencialmente proporcional ao objetivo, e na rede Bitcoin o objetivo aumenta com o tempo (NAKAMOTO, 2008).

2.4 Funções Moderamente Difíceis

O artigo de Nakamoto cita como origem para o *Proof-of-work* o trabalho de Back (2002), que por sua vez cita Dwork e Naor (1993). Nesse último artigo, os autores discutem uma forma de combater spam utilizando uma taxa para envio de e-mails que não atrapalhe a experiência do usuário, mas que dificulte o envio de mensagens em massa. A solução foi exigir que o computador prove que exerceu trabalho ao executar uma função que não seja trivial, mas também não seja impossível, ou seja, uma *Proof-of-work*.

Os autores Dwork e Naor (1993) chamaram essas funções que não são triviais mas ao mesmo tempo não são intratáveis de **funções moderamente difíceis**, propondo um novo campo de estudo e sugerindo que mais pesquisa na área seja realizada. Exemplos de aplicações que podem se beneficiar com o avanço dessa área são o bloqueio temporário de recursos e como prova de realização de esforço, como discutido na seção 2.3. Naor (2003) propõem que funções moderadamente difíceis podem ser usadas como uma forma de criptografia baseada em tempo[2], utlizando funções essencialmente sequenciais, ou seja, que não podem ser paralelizadas. Assim, a estimativa de tempo para a quebra de um problema é aproximadamente o mesmo independente do poder computacional do agente.

2.5 Criptografia Assimétrica

Em 1976, o trabalho de Diffie e Hellman (1976) foi o primeiro a introduzir publicamente[3] o conceito de criptografia assimétrica. Nessa forma de criptografia, existe uma chave para fazer a encriptação e outra diferente, porém relacionada, para fazer a decriptação. Além disso, é computacionalmente impraticável determinar a chave de decriptação utilizando a chave de encriptação.

Em um sistema de chaves-públicas, como no algoritmo RSA (RIVEST; SHAMIR; ADLEMAN, 1978), se uma das chaves for usada para encriptação, a outra será usada para decriptação, o que gera duas grandes vantagens: a primeira é que enviar mensagens cifradas sem precisar compartilhar um segredo; a segunda é se torna possível assinar mensagens para garantir sua origem e autenticidade.

O passo-a-passo do funcionamento básico é o seguinte (STALLINGS, 2017):

[2] Seção 2.6

[3] Stallings (2017) afirma que o conceito de cripgrafia assimétrica já era discutido em trabalhos anteriores que não foram divulgados publicamente.

a) Cada usuário gera um par de chaves;

b) Cada usuário deixa uma das chaves disponível para qualquer um (chave pública) e mantém a outra em segredo (chave privada);

c) Se Bob deseja mandar uma mensagem confidencial para Alice, Bob encripta a mensagem com a chave pública de Alice;

d) Quando Alice recebe a mensagem cifrada, ela decifra usando sua chave privada. como só ela possui a chave privada, só ela pode decifrar;

e) Se Bob deseja mandar uma mensagem para Alice, de modo Alice tenha certeza que Bob é o autor, ele cifra a mensagem usando sua chave privada;

f) Quando alice recebe a cifra, ela decripta usando a chave pública de Bob. Ou seja, somente Bob poderia ter escrito a mensagem.

A Figura 1 representa esse processo. Na imagem (a), Bob encripta o texto X com a chave pública PU_a de Alice e um algoritmo de encriptação assimétrica E, gerando a cifra Y, tal que $Y = E(X, PU_a)$. Alice decripta Y com sua chave privada PR_a e um algoritmo de decriptação D, de modo que $D(Y, PR_a) = X$. Na imagem (b), Bob encripta o texto X com a chave privada PR_b de Alice e um algoritmo de encriptação assimétrica E, gerando a cifra Y, tal que $Y = E(X, PR_b)$. Alice decripta Y com a chave pública de Bob PU_b e um algoritmo de decriptação D, de modo que $D(Y, PU_b) = X$.

Tradicionalmente, esse sistema pode ser usado para troca de mensagens secretas e para gerar certificados digitais.

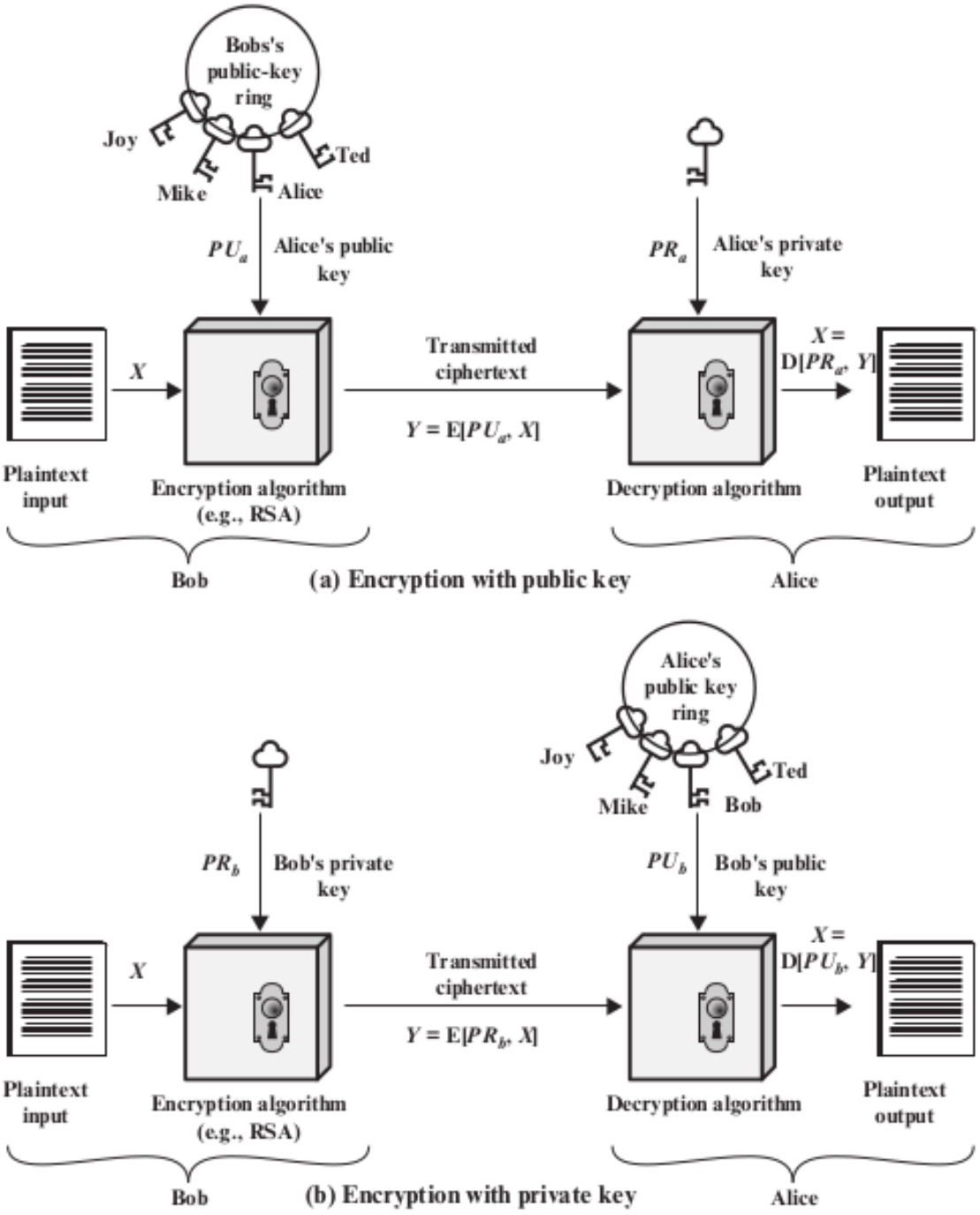
2.6 Criptografia Baseada em Tempo

Um problema interessante na área de criptografia é o estudo sobre como criar um meio de bloquear o acesso à uma informação durante um período de tempo t, e após esse período a informação ficar disponível sem precisar de uma ação da pessoa que bloqueou. O problema também é descrito as vezes como "Mandar uma mensagem para o futuro".

Algumas abordagens foram propostas para lidar com esse problema. Uma linha de pesquisa propõe modelos que assumem uma terceira parte confiável, responsável por liberar chaves de descriptografia no momento correto (RIVEST; SHAMIR; WAGNER, 1996; BONEH; BOYEN; GOH, 2005; CHEON et al., 2008). Outra linha de pesquisa utiliza a ideia de que o receptor da mensagem precisa resolver um problema computacional expressivo, mas não impossível, para desvendar a cifra (BONEH; NAOR, 2000; UNRUH, 2015). A dificuldade do problema define o tempo aproximado para a cifra ser revelada, e os problemas em si são escolhidos de forma que o esforço computacional não pode ser paralelizado.

Liu et al. (2018) define que para ter uma encriptação baseada em tempo completa, é preciso atender simutaneamente os 3 critérios:

Figura 1 – Demonstração do uso de sistema de chaves públicas.



Fonte: (STALLINGS, 2017)

a) **Não interatividade**: o emissor da cifra não é necessário para a decriptação;

b) **Indepedência**: Não deve existir dependência de uma terceira parte confiável. Ou seja, o emissor não precisa depender de um terceiro confiável para guardar as chaves de decriptação até o momento da abertura da cifra chegar.

c) **Economia de recursos**: Interessados em revelar a cifra não devem ser forçados a gastar recursos computacionais para esse objetivo. Isso significa que um receptor

> deve somente esperar até o momento t planejado de liberação da cifra para descobrir a mensagem, e todos os interessados devem descobrir a mensagem aproximadamente ao mesmo tempo independente do poder computacional.

Liu et al. (2018) propõe uma solução que teoricamente atende os três requisitos. Os autores introduzem a ideia de um *relógio computacional de referência*, baseado no estado de uma computação distribuída pública e constantemente iterativa. Como instância desse relógio, os autores utlizam a rede Bitcoin[4]. Somado com o relógio de referência, é utlizado um esquema de criptografia de testemunha (GARG et al., 2013).

[4] Seção 2.3

3
Proposta

Seguindo os princípios propostos por Gritzalis (2002) descritos na seção 2.1, este capítulo apresenta uma arquitetura distribuída para e-voting. O objetivo dessa arquitetura é que os participantes da eleição dependam o mínimo possível de uma autoridade central para a realização do pleito. A eleição deve ser gerenciada de maneira distribuída, mesmo entre partes adversárias, como proposto na Tabela 1:

Tabela 1 – Tranferência de responsabilidade em um modelo descentralizado.

Ação	Responsável no modelo centralizado	Responsável no modelo descentralizado
Gerenciar eleitores	Autoridade central	Autoridade central
Gerenciar candidatos	Autoridade central	Eleitores, candidados, observadores
Lançar votos	Eleitores	Eleitores
Armazenar votos	Autoridade central	Candidados, observadores
Realizar contagem	Autoridade central	Candidados, observadores
Auditar o sistema	Eleitores, candidados, observadores	Eleitores, candidados, observadores

Fonte: Autor.

Na proposta a seguir, são definidos três tipos de papéis exercidos pelos nós da rede:

Eleitores: são os nós responsáveis por emitir os votos. Seguindo o Princípio do Isomorfismo (Seção 2.1.1), os eleitores devem estar autenticados para a emissão do voto. Deve ser possível para o eleitor garantir que o seu voto foi computado, porém não deve ser possível que ele gere provas do conteúdo do próprio voto (Princípios da Verificabilidade e da Incoercibilidade, respectivamente). Ao final da eleição, os eleitores devem ser capazes de verificar e auditar a contagem de votos.

Hosts: são os nós que que armazenam os votos. Os candidatos (ou partidos) devem obrigatoriamente exercer o papel de hosts. Assumindo que uma eleição possui, no mínimo,

dois candidatos, então devem existir pelo menos dois hosts. Outros indivíduos, entidades ou observadores com interesse em ser hosts também podem exercer o papel.

Autoridade de Certificado: Nó responsável somente por emitir e verificar certificados digitais. É a entidade central (fisicamente pode ser distribuída) com poder reduzido no que tange a emissão e contagem de votos. É um nó de consulta de identidade e autorização para os outros nós. Antes da eleição, é preciso que todos os futuros participantes (eleitores, candidatos, demais interessados) tenham chaves públicas cadastradas na Autoridade de Certificado. Ao longo do trabalho, será abreviado para AC.

Dois grandes desafios surgem com esse modelo:

1. Como os hosts devem entrar em consenso sobre o armazenamento de votos?
2. Qual o mecanismo que impedirá os hosts de ler os votos antes do fim da votação?

O primeiro problema pode ser abordado usando a literatura sobre consenso em sistemas distribuídos, inevitavelmente capturando suas vantagens e desvantagens. Importante ressaltar que nesse modelo, apesar do voto ser secreto, os atores envolvidos devem estar devidamente autenticados, mesmo que existam operações que protejam a identidade do usuário, o completo anonimato é indesejável para a solução, pois quebraria o princípio da igualdade entre os eleitores[1]. Isso faz com que as mensagens trocadas estejam, de uma forma ou de outra, autenticadas. Ou seja, dentre os modelos de falha possíveis em sistemas distribuídos, o objetivo é que o modelo proposto seja tolerável a falhas bizantinas com autenticação[2].

O segundo problema é mais complexo. Em outros contextos, quando um ator precisa armezenar uma informação em um repositório não confiável, essa informação é criptografada; para decifrá-la, é necessária uma ação do próprio ator ou de um outro ator autorizado por ele. O problema surge no contexto de uma eleição, quando em determinado momento do processo eleitoral, o voto precisa ser desassociado do eleitor. É preciso, nesse caso, que após um momento t, a criptografia que protege o voto seja quebrada sem ação do ator que encriptou, que nesse contexto é o eleitor. De maneira ideal, é preciso que esse sistema de encriptação cumpra três propriedades simultaneamente:

a) O eleitor não precisa estar presente para a decriptação;
b) O sistema não pode depender de um terceiro que guardará as chaves de decriptação;
c) As partes interesadas em decriptar a cifra não podem ter uma vantagem significativa uma sobre as outras de acordo com o seu poder computacional. Ou seja, todas as partes devem poder decifrar o voto aproximadamente ao mesmo tempo e nunca antes do tempo t.

[1] Ver a seção 2.1.7
[2] Ver a seção 2.2

Portanto, é preciso que essa encriptação seja baseada em tempo[3]. Para esse modelo, é definido que o mecanismo de encriptação é baseado no trabalho de Liu et al. (2018), pois as três propriedades acima são análogas às propriedades da proposta dele. Também será apresentado uma alternativa baseada em funções moderamente difíceis (Ver a seção 2.4).

3.1 Regras, Restrições e Notações

Para servir como base para a construção e entendimento do modelo proposto, é preciso definir o conjunto de regras e premissas da proposta:

a) O sistema consiste em um conjunto distribuído de nós;

b) A notação para a dupla de chaves assimétricas de um nó n é (PK_n, PR_n), onde PK_n é a chave pública e PR_n é a privada. A notação $E(m, k) = c$ significa que a mensagem m é criptografada com a chave k e gera uma cifra c. A notação $D(c, k) = m$ significa que a cifra c é descriptografado com a chave k e gera a mensagem m;

c) Os nós podem participar de um evento chamado Eleição. Durante uma Eleição existem as seguintes regras:

   – Uma eleição é composta por um questionamento q, que pode ser respondido com uma proposta p. O conjunto de propostas é definido por P;
   – Cada host pode emitir somente uma proposta p;
   – Cada eleitor pode emitir somente um voto v, em demonstração de apoio à somente uma proposta qualquer. A notação para extrair uma proposta de um voto é $fv(v) = p_n$. O conjunto de votos computados de uma eleição é V;
   – O número de candidatos deve ser menor ou igual ao o número total de hosts;
   – O número de candidatos deve ser maior que 1.

d) Os nós obrigatoriamente devem realizar as etapas da votação através de uma rede de computadores (i.e., online);

e) Os nós utilizam o protocolo NTP (Network Time Protocol) para sincronizar os relógios;

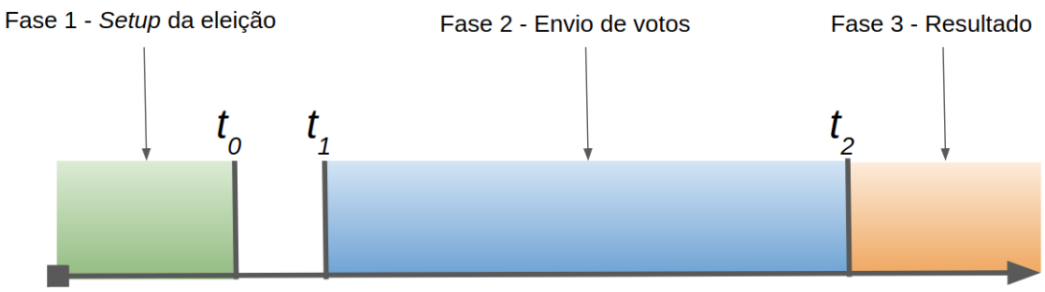
f) As trocas de mensagens entre os nós são feitas com criptografia ponta-a-ponta.

3.2 Primeira Fase: setup da eleição

Uma eleição é comumente divida em três etapas (Figura 2): uma preparação, onde as características da eleição são definidas; o período de lançamento dos votos, que ocorre um tempo depois do encerramento da primeira fase; a contagem de votos, que pode ocorrer imediatamente após o fim das eleições. As divisões entre esses períodos de tempo serão chamadas de t_0, t_1, t_2.

[3] Ver a seção 2.6

Figura 2 – Tmeline de uma eleição.



Fonte: Autor.

Etapa 1 (Criação da eleição): Uma eleição é criada a partir de qualquer nó da rede. A eleição é uma 6-upla e:

$$e = \langle q, t_0, t_1, t_2, f, bs \rangle$$

O primeiro elemento é o questionamento q. O questionamento representa a pergunta que deve ser respondida com a eleição. Exemplos de questionamentos são *"Qual chapa deve administrar o centro estudantil?"* ou *"O Reino Unido deve sair da União Europeia?"*. O segundo é um *timestamp* futuro t_0, que indica um prazo máximo para os nós enviarem propostas. O terceiro é um *timestamp* futuro t_1 que indica o início do recebimento de votos. O quarto é outro *timestamp* futuro t_2, que indica um prazo máximo para o recebimento de votos. Apesar de ser um sistema distribuído, não é preciso utilizar um esquema de relógio lógico de Lamport (1978), pois há uma diferença na sincronização dos relógios tolerável de tal modo que o uso do NTP é suficiente.

O quinto elemento é a função $f : V \longrightarrow P$ que define a proposta vencedora baseada no conjunto de votos. Por exemplo, seja A o conjunto de votos em uma proposta p_n. Ou seja, $v \in A$ se $fv(v) = p_n$. Se for decidido que a proposta vencedora é a que possuir maioria simples, então $f(V) = p_n$, se $|A| > 0.5|V|$.

O sexto elemento é um parâmetro $bs \in \mathbb{N}$ e $bs < |E|$, sendo E o conjunto de eleitores, que indica o tamanho dos blocos de divulgação. Os hosts devem divulgar os votos em grupos, para equilibrar o sigilo e a verificabilidade. Esse processo é detalhado na seção 3.3.2.

O nó iniciador deve tornar e público para os outros nós. Para garantir a autenticidade da proposta, o nó deve assinar a mensagem.

Etapa 2 (Cadastro de propostas e hosts): Ao tomar conhecimento de uma eleição e, um nó n qualquer pode emitir uma proposta e se tornar um candidato antes do tempo t_0. Para isso, além da proposta p_n, ele precisa definir um endereço de host h_n de onde fará a recepção e o armazenamento de votos. O candidato, então, deve tornar público p_n e h_n e assinar a

mensagem contendo os dois valores para garantir sua origem. Outros interessados em manter um nó armazenador (i.e., host) também podem realizar o mesmo processo, mas sem gerar a proposta.

3.3 Segunda Fase: Emissão e Armazenamento dos Votos

Etapa 1(Geração de identificação): Com a chegada de t_0, a primeira fase é encerrada e neste momento, um nó eleitor n possui as informações sobre a eleição com uma cópia da estrutura e, uma lista de propostas $p_1, p_2 ... p_n$ e uma lista de endereços de hosts $h_1, h_1 ... h_k$, tal que $n \leq k$. O eleitor precisa então gerar uma identificação temporária que cumpra os seguintes requisitos:

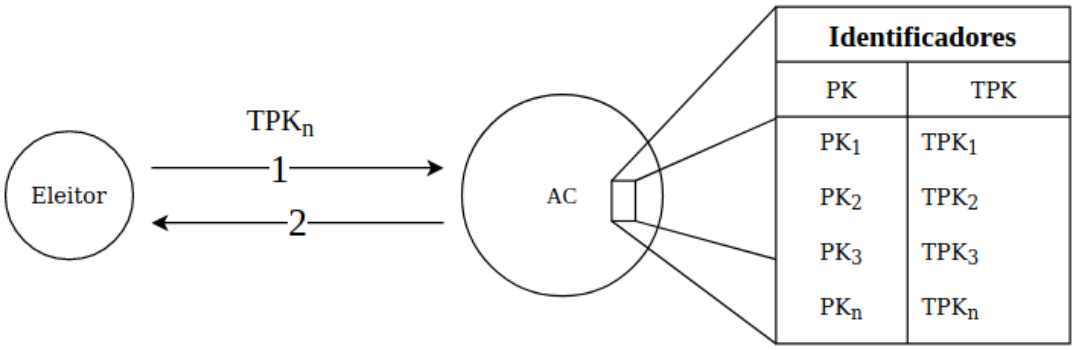
a) Deve ser possível para o AC verificar que a identificação foi gerada por um eleitor válido e que essa identificação é a única gerada por aquele eleitor para aquela eleição. Ou seja, cada eleitor deve possuir uma e somente uma identificação;

b) Não deve ser possível para os outros participantes da eleição associar uma identificação a um eleitor;

c) Qualquer um deve poder verificar se uma identificação é válida para aquela eleição.

Para cumprir com o requisito c), o eleitor deve gerar um dupla de chaves $I = (TPK_n, TPR_n)$ que servirá como identificação temporária e cadastrar no AC TPK_n, (mensagem 1 na Figura 3). O AC deve verificar se já existe alguma identificação daquele eleitor para aquela eleição, e responder ao eleitor se o cadastro foi feito com sucesso. (mensagem 2 na Figura 3).

Figura 3 – Etapa 1: envio das chaves temporárias para AC.

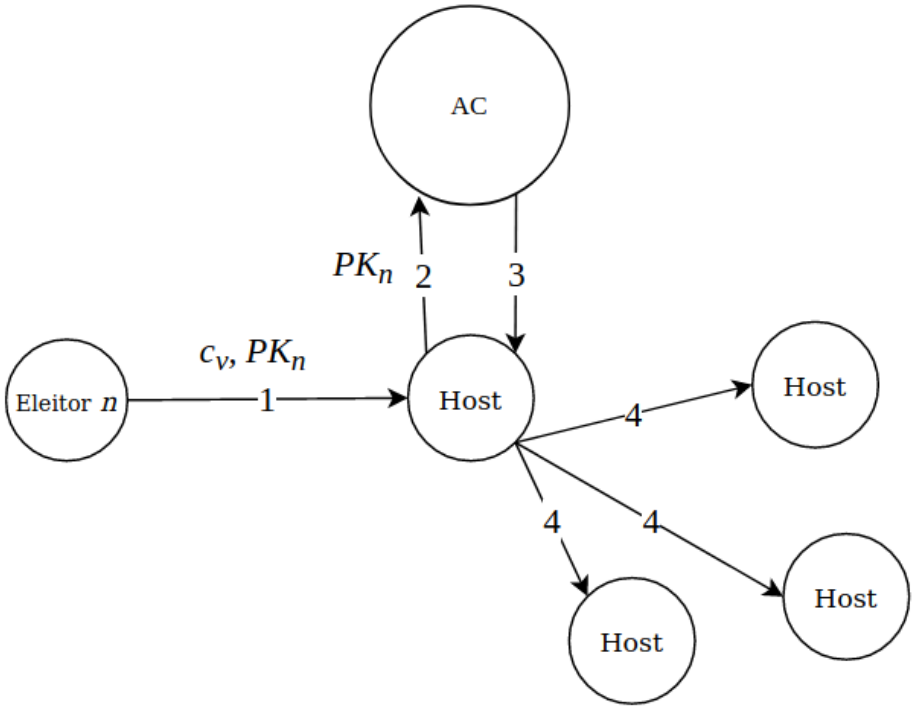


Fonte: Autor.

Etapa 2 (Envio do voto): Após o momento t_1, o eleitor poderá emitir seu voto v, encriptografado e seguindo os requisitos descritos no começo deste capítulo. Para isso, ele deve usar um esquema de criptografia baseada em tempo, fundamentado no trabalho de Liu et al. (2018). A cifra c_v representa a crifra do voto após a encriptação.

O eleitor deve enviar c_v, assinada com I, para um host h_i qualquer da eleição (mensagem 1 na Figura 4). O host deve verificar com AC se a chave pública de I é válida (mensagem 2 na Figura 4). Caso não seja, a mensagem deve ser ignorada. Caso seja válida, o host deve compartilhar com todos os outros hosts a cifra c_v assinada, através de um algoritmo de consenso mensagens 4 na Figura 4).

Figura 4 – Etapa 2: sequência de troca de mensagens para o envio de votos.



Fonte: Autor

3.3.1 Consenso

Como discutido no início deste capítulo, é preciso que o consenso entre os hosts seja tolerável a falhas bizantinas com autenticação. Além disso, serão definidos os requisitos que o algoritmo deve cumprir no contexto da eleição distribuída.

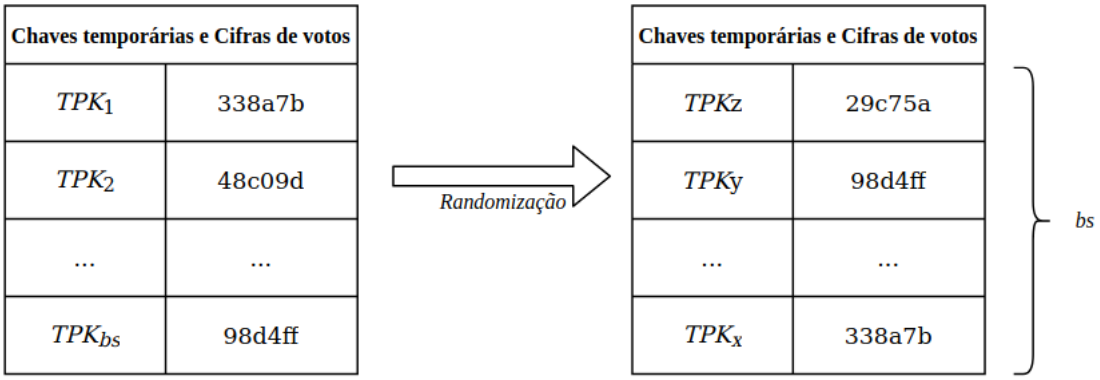
O primeiro requisito é que se o eleitor mandar uma cifra assinada c_v para um host h_i, ele deve receber a confirmação de recebimento por outro host $h_{j,j\neq i}$. Isso é feito para garantir ao eleitor que houve um consenso e que a mensagem foi distribuída. O segundo requisito é que cada vez que um host receber uma cifra assinada c_v, ele deve verificar a validade da chave pública da assinatura com AC. Além disso, o host precisa verificar esse é o primeiro voto do eleitor ou se já existe algum outro voto acordado pelo consenso. Esse requisito serve como verificação do host de que a mensagem se originou de um eleitor válido. Qualquer algoritmo de consenso que cumpra esses requisitos, por exemplo o PBFT, pode ser usado nessa fase.

3.3.2 Confirmação e Exibição dos votos

Após a realização do consenso, algum host $h_{j\neq i}$ deve enviar uma confirmação de recebimento para o eleitor. Quando o eleitor recebe a confirmação de um host $h_{j\neq i}$, a operação de envio do voto é concluída.

Após uma determinada quantidade bs de cifras de votos recebidas, os hosts devem divulgar publicamente essas cifras em uma tabela, junto com os valores da chave pública TPK que o acompanham. Os valores das chaves e das cifras não podem estar diretamente ligados um com outro e portanto a ordem das colunas devem ser randômica (ver a Figura 5). A divulgação dessas tabelas dessa maneira é feita por dois motivos que buscam atender o princípio da verificabilidade. O primeiro é que as cifras devem ser públicas antes de serem abertas. Momentos antes da cifra ser decifrada, todos poderão ter uma cópia do conjunto de votos. O segundo motivo é que se o conjunto de cifras está acompanhado de um conjunto de chaves públicas temporárias, mesmo que em ordem aleatória, é possível para qualquer um checar com o CA se as chaves são válidas.

Figura 5 – Etapa 2: Randomização da tabela com cifras de votos.



Fonte: Autor

3.3.3 Resumo da Fase

A sequência de ações completa dessa fase é:

a) O eleitor gera $I = (TPK_n, TPR_n)$;

b) O eleitor envia uma mensagem de cadastro de I para AC;

c) AC envia uma mensagem de confirmação para o eleitor;

d) O eleitor envia a mensagem c_v, assinada com I para um host h_i;

e) O host verifica com AC se TPK_n pertence a uma identidade válida;

f) Caso a resposta seja positiva, o host manda em *groupcast* c_v para os outros hosts;

g) Todos os outros hosts $h_{j\neq i}$ realizam a verificação de TPK_n com AC;

h) Pelo menos um host $h_{j\neq i}$ manda a confirmação de recebimento para o eleitor e para h_i;

i) O eleitor espera receber a confirmação de pelo menos um host $h_{j\neq i}$ até um *timeout* fixo ou como parâmetro especificado na criação da eleição. Caso isso não ocorra, o eleitor repete o envio para um host $h_{j\neq i}$

j) h_i espera receber a confirmação de pelo menos um host $h_{j\neq i}$ até o *timeout* especificado para aceitar o voto. Caso isso não ocorra, o voto é descartado.

k) Para qualquer host, após uma quantidade bs de votos aceitos, é publicada uma tabela com uma coluna de TPK e uma coluna de c_v, ambas em ordem randômica.

3.4 Terceira Fase: Contagem dos Votos

Após a chegada do *timestamp* t_2, a coleta de votos deve ser encerrada. Nesse momento, cada host já divulgou tabelas com as cifras dos votos em consenso, e todos os interessados podem ter acesso. Caso t_2 chegue e a quantidade de votos é menor que bs, as hosts divulgam uma tabela menor. Seguindo o proposto no início do capítulo, após o momento t_2, as cifras são desfeitas e a proposta vencedora pode ser extraída do conjunto de votos utilizando a função f definida no momento da criação da eleição. Portanto, a contagem não é responsabilidade de somente um ator, a verificação e contagem dos votos pode e deve ser realizada por múltiplas entidades.

3.5 Alternativa para encriptação do voto

Nesta seção, é proposta uma alternativa para criptografar o voto utilizando criptografia assimétrica e funções moderamente difíceis.

Suponha que Alice precise mandar uma mensagem para Bob e Carlos, mas a mensagem deve ser aberta um pouco depois do momento t, mas nunca antes. Bob e Carlos são adversários, então um não quer que o outro veja as mensagens antes de t. Além disso, Alice também não deve ser necessária para decifrar as mensagens.

Alice poderia utilizar *Timed Commitments* (BONEH; NAOR, 2000) para cifrar a mensagem e mandar para os dois. O problema é resolvido para somente uma mensagem, de uma única pessoa. Mas, e se várias pessoas quisessem mandar mensagens para Bob e Carlos, seguindo as mesmas restrições? O número de mensagens poderia ser maior que o poder de parelelização de Bob e Carlos, e assim não é garantido que todas vão ser decifradas pouco tempo após t.

Uma solução seria garantir que a chave de descriptografia de todas as mensagens fosse a mesma. Assim, no momento que uma fosse quebrada, todas também seriam sem custo adicional. Para realizar isso sem que Alice e os outros emissores precisem trabalhar de forma conjunta, é proposto que Bob e Carlos criem, cada um, um par de chaves pública-privada temporárias. Essas

chaves devem ser geradas de modo que, a partir de uma, é possível descobrir a outra computando uma função moderamente difícil e com um limite inferior de tempo igual a t.

Alice e os outros emissores devem então encriptografar suas mensagens duas vezes: uma usando a chave pública temporária de Bob e na outra usando a de Carlos. Dessa maneira, nenhum dos dois pode abrir a mensagem antes de descobrir a chave privada temporária um do outro. Como eles são adversários, eles não vão compartilhar as chaves privadas antes do tempo t.

Para uma eleição distribuída, podemos interpretar os emissores e Alice como os eleitores, Bob e Carlos como candidatos e o tempo t como o momento da divulgação dos votos. Assim, os próprios candidatos (que seriam os hosts) podem guardar os votos sem quebrar o sigilo. Um entrave para essa proposta é saber se é possível gerar as chaves dessa maneira, parametrizando o tempo de quebra com t e resiste à computação paralela.

4

Considerações Finais

Foi apresentado no capítulo 3 uma proposta de arquitetura para uma eleição eletrônica distribuída. O objetivo dessa arquitetura é estudar a viabilidade de um sistema com a responsabilidade de uma entidade central reduzida e os requisitos tecnológicos necessários para isso.

No trabalho de Hjálmarsson et al. (2018), o eleitor pode recuperar seu voto utilizando um identificador, o que torna o sistema mais auditável. O presente trabalho mostrou que a literatura não considera essa ação recomendável, pois impossibilita totalmente um dos pricípios da eleição segura e permite que o eleitor possa ser coagido (Ver a seção 2.1.3). Também é discutido neste trabalho como o voto pode ser armazenado secretamente até o momento do fim da eleição, o que não foi apresentado em Hjálmarsson et al. (2018).

A proposta deste trabalho também apresenta vantagem na proteção contra coerção sobre a arquitetura de Hardwick et al. (2018). O autor apresenta um sistema em que, para que para evadir coerção, o eleitor poderá mudar seu voto após o lançamento. Isso não garante a segurança do eleitor e vai contra a recomendação da literatura.

4.1 Problemas e possíveis soluções

Ao destribuir o processo eleitoral, vantagens são adquiridas mas novos problemas surgem. Nesta seção, alguns destes problemas sãp descritos e possíveis soluções discutidas.

4.1.1 AC Malicioso

A dependência potencialmente perigosa de uma entidade central foi uma das motivações deste trabalho. A entidade central foi reduzida, então, a uma Autoridade de Certificado, responsável somente gerenciar as identidades dos participantes. Mesmo assim, caso a AC seja comprometida, a eleição inteira pode se tornar não confiável. Caso a AC seja malisiosa e queira beneficiar um

host específico, no momento que esse host consulta a AC sobre a validade de uma chave pública temporária[1] ele pode devolver a identidade original do eleitor. Além disso, se o host mandar as tabelas com as colunas não randomizadas para AC, ele poderá relacionar uma tabela com um voto.

Nesse caso, a mitigação de ameaça é semelhante ao de uma entidade central em uma eleição tradicional. Deve ser garantido aos eleitores e partidos participar de processos de auditoria da AC e todas as ações que não possam comprometer a identidade dos eleitores devem ser monitoradas e geradas *logs*. Mais especificamente, deve-se garantir que a comunicação da AC com os hosts deve ser somente as descritas na arquitetura e que não haverá canais paralelos de comunicação.

4.1.2 Coercibilidade na Eleição Online

Na apresentação do princípio da incoercibilidade[2], é dito que é preciso prover o sigilo da emissão do voto até mesmo fora do sistema. Gritzalis (2002) afirma que um problema inerente aos sistemas de votação eletrônicos em que o voto pode ser emitido pela Internet é que não há garantia de que o eleitor não será coagido no momento do lançamento. Ou seja, mesmo que exista um sistema eletrônico perfeito, o eleitor pode ser coagido fora dele.

A solução para esse problema também é a mesma de sistemas eleitorais tradicionais: uma estrutura física com uma cabine de votação que isole o eleitor no momento do lançamento do voto. No contexto de eleições menores, em que o resultado da eleição não é tão impactante ou o poder de coerção dos interessados é reduzido, esse quesito pode ser relaxado. Entretanto, é importante ressaltar que este sempre será um ponto de falha.

4.1.3 Consenso em Sistemas Bipartidários e Coligações

Um dos pontos cruciais em sistemas distribuídos sem autoridade central é o consenso. O algoritmo PBFT (*Practical Byzantine Fault Tolerance*) de Castro, Liskov et al. (1999) garante consenso em até $\frac{n-3}{3}$ nós maliciosos, sendo n o número total de nós. Ou seja, para uma rede tolerar pelo menos um nó malicioso usando PBFT, é preciso no mínimo quatro nós na rede. O *Proof-of-Work* no Bitcoin garante consenso desde que uma entidade não controle mais do que 50% do poder computacional da rede. Para esse algoritmo, é possível uma rede com dois nós, mas eles precisam manter um equilíbrio constante de poder computacional, o que é altamente improvável quando os nós são adversários.

Esses limites impossibilitam o consenso para sistemas completamente bipartidários. Além disso, supondo (um cenário completamente possível) que em um sistema multipartidário os partidos possam traçar alianças, de modo que se determinado partido perceber que não terá

[1] Etapa 2 da seção 3.3
[2] Ver a seção 2.1.3

chance de vencer, ele oferecerá suporte a um outro partido com mais chances. Inevitavelmente os dois partidos com mais chances de vencer formarão duas grandes coligações, reduzindo o problema para um sistema bipartidário. O consenso nesse caso só é possivel se for garantido que haverá entidades completamente imparciais na rede.

4.1.4 Complexidade da solução

O uso de criptografia assimétrica, criptografia baseada em tempo, algoritmos de consenso e funções moderamente difíceis faz com que o entendimento da arquitetura não seja trivial. Isso fere o princípio da simplicidade[3]. Por outro lado, mecanismos de segurança são inevitavelmente complexos (STALLINGS, 2017). Esse problema pode ser minimizado através de manuais e textos descritivos que apresentem uma visão global dos processos eleitorais.

4.2 Trabalhos futuros

O primeiro passo faz parte dos requisitos listados na primeira seção do capítulo 2. Para atender os princípios da transparência, segurança e verificabilidade, é necessário que profissionais da área e demais pesquisadores façam análises críticas e independentes deste trabalho, descobrindo possíveis falhas não mapeadas e validando processos e tecnologias.

O segundo passo é implementar um sistema baseado na arquitetura proposta, e realizar análises de desempenho e segurança. A implementação também deve ser auditada por um outro pesquisador. Também nessa implementação, deve ser analisada a viabilidade da criptografia baseada em tempo proposta por Liu et al. (2018).

Outro possível trabalho futuro é o estudo para verificar se existe uma forma melhor de anonimizar o eleitor. No cenário ideal, é preciso gerar um identificador que atenda simultaneamente os seguintes requisitos:

a) O identificador foi gerado a partir do conjunto de eleitores válidos, sem especificar qual eleitor;

b) Um eleitor não pode gerar dois identificadores.

Deve ser estudado se é possível atender ambos os requisitos simultaneamente.

4.3 Conclusão

Mais estudos teóricos sobre o desenvolvimento de sistemas de e-voting distribuídos ainda devem ser feitos antes de uma implementação. Em especial, melhores soluções relacionadas ao sigilo do voto e incoercibilidade devem ser propostas. Apesar da divisão de responsabilidades, a eleição ainda pode ser pode ser manipulada caso a entidade central seja maliosa.

[3] Seção 2.1.12

A criptografia baseada em tempo apresenta potencial como solução para o armazenamento distribuído de votos, porém é preciso que sejam feitos mais estudos sobre viabilidade e desempenho.

A literatura sobre o consenso da rede de hosts já está estabelecida, porém é preciso ressaltar que em sistemas bipartidários o consenso é prejudicado, tanto em soluções mais tradiconais quanto em algoritmos de consenso baseados em blockchain. Apesar do potencial da arquittura distribuída, muito ainda precisa ser feito.

Referências

ARANHA, D. F.; GRAAF, J. van de. The good, the bad, and the ugly: Two decades of e-voting in brazil. *IEEE Security Privacy*, v. 16, n. 6, p. 22–30, 2018. Citado 2 vezes nas páginas 9 e 11.

BACK, A. Hashcash - a denial of service counter-measure. 2002. Disponível em: <http://www.hashcash.org/papers/hashcash.pdf>. Citado na página 19.

BONEH, D.; BOYEN, X.; GOH, E.-J. Hierarchical identity based encryption with constant size ciphertext. In: SPRINGER. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. [S.l.], 2005. p. 440–456. Citado na página 20.

BONEH, D.; NAOR, M. Timed commitments. *Advances in Cryptology — CRYPTO 2000*, p. 236–254, 2000. Disponível em: <https://doi.org/10.1007/3-540-44598-6_15>. Citado 2 vezes nas páginas 20 e 30.

CASTRO, M.; LISKOV, B. et al. Practical byzantine fault tolerance. In: *OSDI*. [S.l.: s.n.], 1999. v. 99, n. 1999, p. 173–186. Citado 2 vezes nas páginas 18 e 33.

CHEON, J. H. et al. Provably secure timed-release public key encryption. *ACM Trans. Inf. Syst. Secur.*, Association for Computing Machinery, New York, NY, USA, v. 11, n. 2, maio 2008. ISSN 1094-9224. Disponível em: <https://doi.org/10.1145/1330332.1330336>. Citado na página 20.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory*, Elsevier, v. 22, p. 644–654, 1976. Disponível em: <https://ee.stanford.edu/~hellman/publications/24.pdf>. Citado na página 19.

DWORK, C.; NAOR, M. Pricing via processing or combatting junk mail. *Advances in Cryptology - CRYPT0 '92*, p. 139–147, 1993. Disponível em: <https://doi.org/10.1007/3-540-48071-4_10>. Citado 2 vezes nas páginas 18 e 19.

DZIEMBOWSKI, S. et al. Proofs of space. In: SPRINGER. *Annual Cryptology Conference*. [S.l.], 2015. p. 585–605. Citado na página 18.

GARG, S. et al. Witness encryption and its applications. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 2013. p. 467–476. Citado na página 22.

GAŽI, P.; KIAYIAS, A.; ZINDROS, D. Proof-of-stake sidechains. In: IEEE. *2019 IEEE Symposium on Security and Privacy (SP)*. [S.l.], 2019. p. 139–156. Citado na página 18.

GRITZALIS, D. A. Principles and requirements for a secure e-voting system. *Computers & Security*, Elsevier, v. 21, p. 539–556, 2002. Disponível em: <https://doi.org/10.1016/S0167-4048(02)01014-3>. Citado 8 vezes nas páginas 9, 10, 13, 15, 16, 17, 23 e 33.

HARDWICK, F. S. et al. E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018. Disponível em: <https://doi.org/10.1109/Cybermatics_2018.2018.00262>. Citado 3 vezes nas páginas 11, 13 e 32.

HJáLMARSSON, F. et al. Blockchain-based e-voting system. In: *IEEE 11th International Conference on Cloud Computing(CLOUD)*. IEE, 2018. Disponível em: <https://doi.org/10.1109/CLOUD.2018.00151>. Citado 3 vezes nas páginas 9, 11 e 32.

KSHEMKALYANI, A. D.; SINGHAL, M. *Distributed Computing: Principles, Algorithms, and Systems*. 1. ed. USA: Cambridge University Press, 2008. ISBN 0521876346. Citado na página 17.

KSHETRI, N.; VOAS, J. Blockchain-enabled e-voting. *IEE Software*, IEEE, v. 35, p. 95–99, 2018. Disponível em: <https://doi.org/10.1109/MS.2018.2801546>. Citado 2 vezes nas páginas 9 e 11.

LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, v. 21, p. 558–565, 1978. Disponível em: <https://doi.org/10.1007/s10623-018-0461-x>. Citado 2 vezes nas páginas 17 e 26.

LAMPORT, L. The weak byzantine generals problem. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 30, n. 3, p. 668–676, 1983. Citado na página 18.

LAUER, T. W. The risk of e-voting. *Electronic Journal of E-government*, Citeseer, v. 2, n. 3, p. 177–186, 2004. Citado 2 vezes nas páginas 9 e 11.

LIU, J. et al. How to build time-lock encryption. *Designs, Codes and Cryptography*, v. 86, p. 2549–2586, 2018. Disponível em: <https://doi.org/10.1007/s10623-018-0461-x>. Citado 5 vezes nas páginas 20, 22, 25, 27 e 34.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Citado 2 vezes nas páginas 18 e 19.

NAOR, M. Moderately hard functions: From complexity to spam fighting. In: SPRINGER. *International Conference on Foundations of Software Technology and Theoretical Computer Science*. [S.l.], 2003. p. 434–442. Citado na página 19.

PHILLIPS eborah M.; SPAKOVSKY, H. A. von. Gauging the risks of internet elections. *Communications of the ACM*, ACM, v. 44, p. 73–85, 2001. Disponível em: <https://dl.acm.org/doi/10.1145/357489.357512>. Citado na página 15.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, ACM New York, NY, USA, v. 21, n. 2, p. 120–126, 1978. Citado na página 19.

RIVEST, R. L.; SHAMIR, A.; WAGNER, D. A. Time-lock puzzles and timed-release crypto. Massachusetts Institute of Technology. Laboratory for Computer Science, 1996. Citado na página 20.

SAMPIGETHAYA, K.; POOVENDRAN, R. A framework and taxonomy for comparison of electronic voting schemes. *Computers & Security*, Elsevier, v. 25, p. 137–153, 2006. Citado na página 13.

SPRINGALL, D. et al. Security analysis of the estonian internet voting system. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2014. (CCS '14), p. 703–715. ISBN 9781450329576. Disponível em: <https://doi.org/10.1145/2660267.2660315>. Citado 2 vezes nas páginas 9 e 11.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. Global edition. Edinburgh Gate, Harlow, Essex CM20 2JE, England: Pearson, 2017. Citado 4 vezes nas páginas 9, 19, 21 e 34.

UNRUH, D. Revocable quantum timed-release encryption. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 62, n. 6, p. 1–76, 2015. Citado na página 20.

ZHENG, Z. et al. An overview of blockchain technology: Architecture, consensus, and future trends. *IEEE International Congress on Big Data (BigData Congress)*, p. 557–564, 2017. Disponível em: <https://doi.org/10.1109/BigDataCongress.2017.85>. Citado na página 18.

ZISSIS, D.; LEKKAS, D. Securing e-government and e-voting with an open cloud computing architecture. *Government Information Quarterly*, Elsevier, v. 28, p. 239–251, 2011. Disponível em: <https://doi.org/10.1016/j.giq.2010.05.010>. Citado 2 vezes nas páginas 11 e 13.