Ranieri Schroeder Althoff

Implementação de um sistema de eleição remoto secreto e verificável

Florianópolis
5 de dezembro de 2018

Ranieri Schroeder Althoff

Implementação de um sistema de eleição remoto secreto e verificável

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE
Programa de Graduação em Ciências da Computação

Orientador: Taciane Martimiano

Florianópolis
5 de dezembro de 2018

Este trabalho é dedicado aos eternos amigos André de Carvalho Ribeiro e Lucas Sant'Helena Nunes, que infelizmente nos deixaram cedo demais, mas sempre trouxeram muitos sorrisos.

AGRADECIMENTOS

Agradeço à minha família, que sempre me proveu todo o suporte e incentivo necessário nessa carreira que está apenas começando.

Agradeço à minha orientadora Taciane e meu professor responsável Jean por todas as conversas e orientações que me ajudaram a moldar este trabalho, e pelo vasto conhecimento compartilhado que me permitiu chegar longe na minha pesquisa.

Agradeço aos professores que tive ao longo da graduação, em especial aos professores Antônio Mariani, Elder Santos, Jerusa Marchi, Matheus Bortolan, Rafael Cancian e Sérgio Peters, com os quais tive mais contato, por todo o aprendizado e inspiração para me tornar um cientista da computação.

Agradeço aos amigos que fiz no movimento estudantil, em especial aos membros do Centro Acadêmico Livre de Computação, com quem conquistei grandes feitos para buscar uma universidade melhor para os estudantes.

Agradeço aos amigos do Caravela Hacker Club, com quem aprendi que sempre há algo novo para se aprender, e que formaram o melhor esquadrão antibombas que eu pude ter.

Agradeço aos peladeiros do futebol quinta-feirino, cujos gols, dribles, chutes tortos e entradas maliciosas me forneceram o alívio cômico e a companhia no bar tão necessários nas semanas mais estressantes.

Por fim, mas não menos importante, agradeço aos incontáveis amigos que fiz até aqui ao longo dessa jornada.

Liberdade é uma palavra que o sonho humano alimenta, que não há ninguém que explique e ninguém que não entenda.
(Cecília Meireles)

RESUMO

Este trabalho visa estudar sistemas de eleição através da internet, em especial o protocolo ADDER (KIAYIAS; KORMAN; WALLUCK, 2006), abordando o histórico dos processos eleitorais, e propor uma implementação do processo eleitoral através de um *website*. A implementação resultante combina o conceito de um registro público de informações (BENALOH, J. D. C., 1987), utilizando o criptosistema de Paillier (PAILLIER, 1999) para prover criptografia dos votos publicados e a divisão de responsabilidade entre autoridades (FOUQUE; POUPARD; STERN, 2000) a fim de promover um ambiente seguro que evita pontos únicos de falha para uma eleição online.

Palavras-chave: criptografia. eleições. privacidade. voto online.

ABSTRACT

The presented work aims to study internet-based election systems, notably the ADDER protocol (KIAYIAS; KORMAN; WALLUCK, 2006), traversing the history of election processes, and to propose an implementation of an election process through a website. The resulting implementation combines the concept of a bulletin board (BENALOH, J. D. C., 1987), while using the Paillier cryptosystem (PAILLIER, 1999) to publish encrypted votes and a threshold authority responsibility (FOUQUE; POUPARD; STERN, 2000) to provide a secure environment that avoids a single point of failure for an online election.

Keywords: cryptography. elections. online voting. privacy.

SUMÁRIO

LISTA DE DEFINIÇÕES

Auditabilidade fim-a-fim

Característica de sistemas de eleição com forte integridade e resistência à violação. Como o nome sugere, um sistema com tal característica pode ser verificado desde o voto até o resultado final pela sua corretude (BENALOH et al., 2015).

Autenticação

Possibilidade da autoridade eleitoral de reconhecer o eleitor e confirmar o seu direito ao voto. Pode ser utilizada uma lista de nomes, um certificado de sociedade, ou um documento eleitoral.

Criptografia homomórfica

Forma de criptografia que permite operações no texto cifrado, gerando um resultado que, quando descriptografado, seja equivalente a aplicar as mesmas operações no texto original (RIVEST; LEDLIE et al., 2002). Sua principal aplicação é manipular informação em ambientes não confiáveis.

Integridade eleitoral

Conjunto de ações para que o resultado das eleições seja de fato a representação do interesse dos eleitores. As entidades do sistema devem contabilizar corretamente cada voto de acordo com a intenção do eleitor, e o resultado final deve ser exatamente a soma destes votos. Além disso, deve haver liberdade para o registro do voto de qualquer eleitor habilitado (ALVIM, 2015).

Mentalidade de segurança

Forma de pensar em soluções e abordar problemas explorando vulnerabilidades de segurança, ou seja, analisar o cenário do ponto de vista de um agressor de forma a detectar falhas antes de serem potencialmente exploradas (SCHNEIER, 2008).

Voto privado

Proteção do eleitor de que seu ato de votar não poderá ser observado por outros participantes do processo. É uma forma de proteção mais fraca que o voto secreto, mas é suficiente para alguns cenários.

Voto secreto

Impossibilidade do sistema de votação fazer qualquer ligação entre votante e voto, impedindo de revelar em que candidato(s) um eleitor votou (DELAUNE; KREMER; RYAN, 2006). Para garantias mais fortes, pode ser desejável que nem mesmo o eleitor possa revelar em quem votou.

1 INTRODUÇÃO

Mesmo com a ubiquidade da internet atualmente, onde fazemos desde compras domésticas até transações bancárias com segurança, a eleição remota é atualmente um tema bastante controverso no mundo da tecnologia da informação. No Brasil, a resistência do tribunal eleitoral de conduzir auditorias irrestritas do sistema eletrônico, acompanhada das vulnerabilidades descobertas no sistema eletrônico, mesmo nos cenários restritos onde auditorias foram realizadas (ARANHA et al., 2012), cria na população uma sensação de inexistência de integridade nas eleições.

O uso de um meio rápido e descentralizado pode trazer diversos benefícios para o eleitor, com destaque para a comodidade de exercer seu voto de qualquer lugar, a redução do tempo de contagem e transporte dos sistemas clássicos de cédulas ou urnas eletrônicas. Por isso, existem tentativas de se utilizar sistemas de votação remota, como nas eleições nacionais da Estônia. No entanto, estas estão aquém dos requisitos de segurança esperados (SPRINGALL et al., 2014), havendo incerteza se é segura o suficiente para a realização de eleições.

Dos diversos sistemas eletrônicos abordados neste trabalho, os mais populares, como sistemas de tabulação de cédulas e as urnas eletrônicas, não estão disponíveis publicamente para uso e estudo, não atendem aos requisitos de segurança de uma eleição, e não passaram por uma extensa bateria de testes de segurança. É consenso na comunidade de segurança digital que segurança por obscurantismo não é eficiente, e se a eleição acontece em uma "caixa preta"não existem meios de verificar a validade dos resulta-

dos.

1.1 JUSTIFICATIVA

Em virtude dos ataques que exploram falhas de segurança, como os *ransomware* presenciados nos anos recentes, observando a falta de conhecimento da população sobre segurança digital, e pensando em uma solução para um dos grandes problemas políticos do país, este trabalho visa apresentar os recentes avanços de segurança digital em uma proposta de sistema seguro que possa ser utilizado para tal.

De forma mais geral, em qualquer votação altamente descentralizada e com uma grande quantidade de eleitores participantes, os sistemas mais ortodoxos e amplamente testados sofrem com problemas de escalabilidade. Considerando que a internet é o meio de comunicação e interação mais ubíquo, a digitalização de sistemas de votação é de grande importância tanto pela acessibilidade permitida, quanto pela segurança provida pelas modernas técnicas de criptografia.

1.2 OBJETIVOS

Este trabalho visa estudar uma solução de eleição remota com alto nível de segurança e privacidade baseada no sistema ADDER (KIAYIAS; KORMAN; WALLUCK, 2006). No seu artigo, o autor citou trabalhos futuros a serem desenvolvidos no sistema que serão abordados no presente trabalho, e serão analisadas as implicações que as alterações possíveis teriam nos requisitos desejados das eleições.

Para isto, serão analisados trabalhos existentes da área, investigando as principais soluções propostas na literatura revisada,

a fim de encontrar restrições e oportunidades de aperfeiçoamento do trabalho. Ao fim, será proposta uma implementação de sistema de eleição online levando em consideração as preocupações destas soluções.

A primeira etapa deste trabalho consiste no levantamento dos requisitos do sistema com relação aos problemas e garantias das eleições online. Em seguida, o estudo e implementação do sistema proposto e as possibilidades de melhoramento em comparação com outros sistemas de objetivo similar.

1.3 ESTRUTURA DO TEXTO

A seção 2.1 apresenta o histórico dos sistemas de eleição desde o início da democracia, e os avanços tecnológicos que os acompanharam. A seção 2.2 faz uma introdução de sistemas criptográficos e eletrônicos.

O Capítulo 3 analisa trabalhos de objetivos similares na literatura, suas considerações e diferenças objetivas.

No Capítulo 4, são descritos cada um dos componentes que formam o sistema de eleição proposto, bem como se dá a interação entre eles quando tecnicamente separados e como garantem os requisitos de segurança esperados.

2 REVISÃO BIBLIOGRÁFICA

2.1 EVOLUÇÃO DOS SISTEMAS DE ELEIÇÃO

Processos de escolha de representantes existem desde que os humanos formaram sociedades, onde nos primórdios os líderes eram os mais fortes e mais saudáveis, alterando com a democracia para um modelo onde os mais aptos são escolhidos pela população. Sistemas de eleição ou votação são aplicados em diversos ambientes, seja para a simples escolha de um representante de turma de colégio, pela tomada de decisão conjunta entre os acionistas de uma empresa, e na escolha dos representantes políticos de um governo ou nação.

2.1.1 Voto por aclamação ou *viva voce*

O primeiro sistema de votação, e tecnologicamente o mais simples, foi implementado nos primórdios da democracia na Grécia Antiga, há quase 3 mil anos. Nele, o eleitor se dirigia aos delegados da eleição e se manifestava em que candidato tinha intenção de eleger (WOLFSON, 1899). Esse tipo de votação tinha múltiplos pontos de falha, muito por sua simplicidade, e aos valores morais e éticos dos responsáveis.

O sistema não possui nenhum mecanismo de privacidade, visto que o voto é público, e não oferece nenhuma resistência à coerção ou venda de voto (BUCHSTEIN, 2010). Por outro lado, esse sistema de votação permite uma perfeita auditoria do resultado, mesmo não havendo garantias de que as autoridades ajam de forma honesta com o resultado da eleição.

A autenticação dos votantes também era problemática na época do uso desse sistema. Como a tecnologia era muito primordial, não haviam grandes registros centralizados de população e formas de comprovar cidadania, de forma que até meados do século XIX os eleitores juravam sobre a bíblia que poderiam votar e o fariam apenas uma vez, sujeitos apenas à sua crença religiosa.

No Brasil, um processo baseado no *viva voce*, onde o votante falava "em segredo" ao juiz responsável foi utilizado durante o período imperial (NICOLAU, 2012). Este curioso sistema de certa forma reduzia o problema da falta de privacidade, mas impedia qualquer forma de auditoria pública do resultado.

O sistema de voto por aclamação não é prático em larga escala, por isso, governos que adotavam este modelo o aplicavam em diferentes níveis (ou graus) de votação indireta. A população de um distrito escolhia o seu representante no colégio eleitoral, que então votaria nos parlamentares em outra instância da eleição. Sistemas indiretos por aclamação foram utilizados na França, Espanha e Portugal, além do Brasil (NICOLAU, 2012).

2.1.2 Urna e cédula de papel

Posterior ao sistema de aclamação, quando governos e eleições se tornaram mais robustos, adotou-se a técnica da cédula de papel para registrar os votos. Nele, o eleitor insere uma cédula em uma urna lacrada e, ao final da eleição, a urna é aberta e os votos são contabilizados. A praticidade e as evoluções desde então fazem com que seja utilizado até hoje em diversas eleições.

Esse sistema provê uma forma simples de sigilo do voto e aprimora a integridade do processo, mas a adição de novos elementos (a urna e a cédula) abriu novas possibilidades de ataques e fraudes. As fraudes elaboradas motivaram o desenvolvimento de

urnas mais seguras, como as urnas de vidro, que permitiam que os eleitores observassem o estado da urna.

É importante aqui salientar a diferença entre privacidade e sigilo neste âmbito. A privacidade do voto ou do eleitor é a sua própria capacidade de revelar ou não sua escolha, como quando se usa uma senha para proteger um arquivo confidencial. Já o sigilo do voto é a impossibilidade de se identificar o eleitor, como uma forma de torná-lo anônimo, e é de certa forma uma segurança mais forte que a privacidade (CONEY et al., 2005).

Existe um balanço entre o sigilo de um voto e a confiabilidade do resultado da eleição, afinal, se não é possível que um eleitor seja identificado pelo seu voto, não se pode verificar que o voto foi corretamente computado na contagem final (JONES; SIMONS, 2012). O dispositivo atualmente proposto para aliviar este dilema é a impressão do voto, cuja implementação fora cogitada para as eleições presidenciais de 2018, mas não prosseguiu.

2.1.2.1 Cédulas impressas

Além da segurança das próprias urnas, o desenho das cédulas também evoluiu para comportar novos níveis de segurança e acessibilidade das votações. Novas cédulas que continham nomes dos candidatos impressos para escolha do eleitor foram instauradas na Austrália, no século XIX.

Embora facilite a escolha por parte do eleitor ao apresentar de antemão os candidatos, em contraste com o preenchimento das cédulas vazias, o desenho da cédula afeta diretamente a intenção de voto e a intuitividade do processo (EVERETT; BYRNE; GREENE, 2006).

A versão impressa exige um nível mínimo de alfabetização do eleitor, uma vez que este precisa identificar qual candidato deve

marcar, que pode restringir a abrangência da eleição, ainda que a cédula utilize símbolos em vez de nomes. Afeta também o custo do processo, pois o governo precisa arcar com os custos da confecção, impressão e distribuição das cédulas.

Para aumentar a privacidade e sigilo, os locais de votação frequentemente utilizam uma cabine onde o eleitor pode, fora da visão pública, preencher sua cédula antes de introduzí-la na urna.

As eleições brasileiras demoraram a adotar as técnicas de sigilo do voto para cédulas. A classe política foi contrária ao voto secreto, argumentando que o cidadão deveria ser responsável por seu voto. Legalmente, o voto secreto chegou a ser proibido (NICOLAU, 2012).

2.1.2.2 Voto encadeado

Um método de fraude bastante elaborado do sistema de cédulas e urnas é o voto encadeado, onde um adversário pode exercer coerção e comprar votos mesmo com a privacidade das cabines de votação.

Esse esquema requer que um adversário que previamente tenha acesso a uma cédula em branco, preencha com o candidato para o qual queira fraudar votos, entregue para um eleitor legítimo e exige deste sua cédula vazia em troca (JONES; SIMONS, 2012).

Assumindo que o mesário entregue apenas uma cédula para o eleitor e exija que uma cédula seja inserida na urna, isso significa que o eleitor depositou a cédula do adversário. O processo é repetido, fraudando votos de forma indetectável por utilizar cédulas legítimas.

Para mitigar este ataque, é possível numerar as cédulas e, antes de inserí-las na urna, conferir se o número a ser inserido é da cédula que foi entregue ao eleitor. A numeração deve ser destacável

e removida antes de inserir na urna, para impossibilitar a distinção de dois votos e o rompimento do sigilo.

2.1.3 Máquinas de alavancas

Máquinas de votação por alavancas foram usadas inicialmente nos Estados Unidos e desenhadas para fazer o processo de votação ser simples e sigiloso. No início, eram o aparato mais tecnológico existente, com muitas partes móveis. Ao se utilizar uma máquina para votar mecanicamente, se evita o problema de interpretação da cédula. Além disso, alguns modelos obrigavam que o eleitor estivesse atrás de cortinas, mantendo a privacidade do seu voto.

Como todo aparato mecânico, as máquinas de alavancas eram suscetíveis a falhas e travamentos das peças, e eram especialmente afetadas justamente pela grande quantidade de partes móveis que as faziam máquinas complexas e avançadas. Uma falha no mecanismo de contagem poderia afetar a contabilização de todos os votos de um candidato.

Nas máquinas com contadores não havia registro individual dos votos, portanto, estas poderiam ser manipuladas por eleitores e técnicos de manutenção para propositalmente falhar na contagem de votos, por vezes de formas indetectáveis (JONES; SIMONS, 2012, p. 42-45). Já nos modelos com impressão em rolo de papel era possível mapear cada voto para cada eleitor, pois eram impressos em ordem (JONES; SIMONS, 2012, p. 26).

As máquinas causaram uma mudança de paradigma das fraudes. A forma mais comum de fraudes com cédulas, de estufar as urnas ou encadear os votos, não era mais possível. No entanto, era igualmente fácil corromper as próprias autoridades eleitorais e, como o eleitor não consegue confirmar que seu voto foi corretamente

contado, as fraudes se tornavam mais silenciosas.

2.1.4 Cartão perfurado

A tecnologia de cartões perfurados foi adaptada para eleições na década de 1960, pela empresa IBM, e aprimorada por professores da Universidade da Califórnia em Berkeley.

Neste método, o cartão perfurado entregue ao eleitor possui espaços para destaque de acordo com cada candidato possível. O eleitor insere o cartão em uma máquina com um catálogo dos candidatos, e deve perfurar o destaque correspondente com o seu voto. A máquina pode ser portátil ou embutida na cabine de votação.

Os problemas com os cartões perfurados apareceram logo após sua introdução em eleições. Pedaços destacados dos cartões se acumulavam nas máquinas e causavam votos indevidos. Este problema causou notável discussão nas eleições dos Estados Unidos em 2000 (LEIB; DITTMER, 2002), sobre se uma área parcialmente destacada deveria ser computada.

Ao contrário das cédulas de papel mais modernas, é mais difícil recontar manualmente os cartões perfurados. Enquanto é fácil verificar uma marca no papel, um destaque incompleto pode ser resultado de hesitação do eleitor que não deseja aquele voto, ou da dificuldade de perfurar o cartão por parte do eleitor que deseja o voto (ROTH, 1998).

Uma derivação dos cartões perfurados, o DataVote traz impresso os nomes dos candidatos no cartão, de forma que o eleitor destaca seu voto diretamente e não é necessária a máquina para destaque dos votos. Desta forma, resolve o problema dos pedaços que atrapalham a realização correta dos votos, e diminui a confusão do eleitor pois não exige o alinhamento exato do cartão na máquina (CRANOR, 2003).

2.1.5 Sensores óticos

O uso de sensores óticos para registro de votos é semelhante aos sistemas de testes padronizados, como o ENEM, e foram usados pela primeira vez em experimentos eleitorais na década de 1960. Apesar da tabulação ser semelhante aos cartões perfurados, é mais intuitivo justamente pelo amplo uso em outras situações (SCHEFFEL, 2002).

Diferentes tecnologias foram usadas como sensores óticos em eleições. Inicialmente, sensores simples contabilizavam marcas no papel da cédula que fossem mais escuras que um dado limite, enquanto modelos mais modernos utilizam algoritmos de reconhecimento de imagens com resoluções altas (JONES; SIMONS, 2012).

Os modelos de máquinas de tabulação poderiam tanto fazer a contagem no local da votação quanto em centrais de contagem de votos, para onde cédulas são levadas após a votação.

Como os outros equipamentos que dependem de *software*, o uso de sensores óticos é suscetível a ataques que ganhem acesso ao sistema operacional da máquina. A cédula é um documento e pode ser utilizada como comprovação de voto numa fraqueza de coerção. Além disso, assim como em cédulas de papel simples, pode sofrer de voto encadeado.

2.1.6 Urnas eletrônicas

Seguindo as tendências da tecnologia, os sistemas eleitorais também foram computadorizados no fim do Século XX. No início do seu uso, a euforia ofuscou a discussão necessária sobre a segurança, privacidade e confiabilidade das urnas eletrônicas, mesmo depois de sua implementação nacional no Brasil e dos problemas encontrados em outros países.

Em análise realizada por Kohno et al. (2004), destaca-se

que as empresas que desenvolveram os primeiros terminais eletrônicos de votação o fizeram de forma que robustez, segurança e confiabilidade das eleições depende inteiramente do *software* sendo executado pelos terminais. Se estes terminais pudessem ser manipulados, seja por eleitores ou desenvolvedores inescrupulosos, ou até pelo sistema operacional utilizado, a precisão dos resultados se torna duvidosa.

Nos Estados Unidos, a lei federal denominada *Help America Vote Act* (HAVA), de 2002, tinha por objetivo incentivar a substituição dos sistemas de cartão perfurado e máquinas de alavancas por equipamentos mais modernos e estabelecer requisitos de acessibilidade aos locais de votação, o que tornava as urnas eletrônicas o único tipo de equipamento que os satisfazia (DAVIS III; THOMAS, 1996).

As urnas eletrônicas são amplamente criticadas por uso incorreto de funções criptográficas e baixa qualidade de código, respaldadas por operarem código proprietário e fechado em geral, e possibilidade de fraude em cartões de configuração, concentrando desconfiança de estudiosos da área (FELDMAN; HALDERMAN; FELTEN, 2006).

A solução então proposta é a do voto impresso, formalmente conhecido como *voter-verifiable audit trail* (rastro de auditoria verificável pelo eleitor). A parte mais importante é a de verificação pelo eleitor: se a máquina não gera uma impressão correta do voto, o papel é destruído mecanicamente e não é inserido na urna de auditoria. Desta forma, o funcionamento do terminal de votação não é mais o único ponto de falha, pois pode ser removido de circulação se estiver produzindo resultados incorretos.

Esse tipo de vulnerabilidade é mais perigosa que as vulnerabilidades dos sistemas tradicionais de cédulas porque são sutis, imperceptíveis da visão do eleitor e das autoridades, e podem acon-

tecer por mero descuido dos responsáveis. Como descoberto por Feldman, Halderman e Felten (2006), um modelo de urna eletrônica utilizada nos Estados Unidos poderia ser manipulada inserindo um cartão de memória com código malicioso em menos de um minuto, situação plausível uma vez que os operadores das urnas frequentemente as operam sem supervisão.

Dentre os poucos relatórios de agentes externos sobre os componentes internos da urna eletrônica, certamente o mais conhecido e completo é o de Aranha et al. (2012), que descreveu diversos problemas de segurança no *software* executado na urna e também na maneira como o órgão responsável modelou os potenciais vetores de ataque ao sistema, por exemplo, a possibilidade de se desfazer o embaralhamento do registro dos votos e obter a ordem com a qual cada voto foi inserido, potencialmente quebrando o sigilo do eleitor.

A situação não é melhor em implementações utilizadas por outros países além do Brasil. Na maior democracia do mundo, a Índia, uma louvável iniciativa do governo instalou milhões de urnas eletrônicas de baixo custo em todo o país, até mesmo nas regiões mais remotas, para maximizar o sufrágio universal. Durante muito tempo, mesmo após a literatura amplamente depreciar as urnas sem voto impresso, o governo insistia que as urnas eram completamente seguras.

Na rigorosa revisão por Wolchok et al. (2010), relata-se que tanto o *software* quanto o *hardware* das urnas utlizadas na Índia pode ser manipulado por um agressor com acesso ao dispositivo. Como as peças são construídas por terceiros, dificilmente um componente com um comportamento malicioso seria detectado na montagem da urna eletrônica, muito menos por um operador leigo. Além disso, o *software* executado também é gravado no *chip* por um terceiro, que poderia instalar uma versão adulterada que dificilmente levantaria suspeitas.

Por fim, pela simplicidade da construção, um agressor poderia tentar substituir uma urna por um modelo adulterado, ou substituir componentes internos da urna por versões maliciosas, considerando que não há uma maneira prática para que operadores verifiquem a autenticidade das urnas sendo utilizadas.

Praticamente todas as democracias do mundo que utilizam urnas eletrônicas e formas similares de terminais de votação, notadamente Venezuela, Países Baixos, Alemanha, Bélgica, Rússia e Equador, além dos já citados Estados Unidos e Índia, já substituíram urnas eletrônicas sem confirmação impressa do voto, agora chamadas de máquinas de primeira geração. O único país que ainda permite máquinas de primeira geração é o Brasil, que está em processo de substituição por máquinas de segunda geração, com confirmação impressa.

2.2 SISTEMAS CRIPTOGRÁFICOS E ELETRÔNICOS

2.2.1 Punchscan

O sistema Punchscan, descrito em (FISHER; CARBACK; SHERMAN, 2006), é um sistema híbrido entre cédulas e dispositivos eletrônicos, que utiliza sensores óticos, implementa auditabilidade fim-a-fim dos resultados e emite um recibo de voto ao eleitor.

Apesar de utilizar um software para contagem de votos, o Punchscan é independente deste. Sua segurança é baseada em premissas criptográficas em vez de depender da segurança do software, como nas urnas eletrônicas, e por isso pode ser executado em sistemas operacionais não confiáveis e ainda manter integridade incondicional.

A cédula do Punchscan é dividida em duas camadas. Na camada superior, os candidatos são listados com um símbolo ao

lado do nome, e abaixo há uma série de furos redondos. Por estes furos, na camada inferior, estão os símbolos correspondentes aos candidatos.

Para efetuar um voto, o eleitor deve encontrar o furo com o símbolo correspondente ao candidato desejado e marcar com uma caneta especial que é propositalmente maior que o furo. O eleitor então separa as camadas, escolhe uma das duas para guardar como recibo e destrói a outra. O recibo é escaneado por um sensor ótico para tabulação do resultado.

A ordem dos candidatos na camada superior é disposta de forma pseudoaleatória, assim como a dos símbolos na camada inferior. Dessa forma, o recibo não contém informação suficiente para determinar o voto: se a camada superior é escolhida, a ordem dos símbolos nos furos é desconhecida; se a camada inferior é escolhida, a ordem dos símbolos referentes aos candidados é desconhecida.

Portanto, o Punchscan é resistente a coerção, uma vez que o eleitor não consegue provar em quem ele votou.

2.2.2 Scantegrity

O sistema Scantegrity, descrito em Chaum, Essex et al. (2008), consiste em uma camada de segurança para sistemas de sensores óticos que implementa auditabilidade fim-a-fim dos resultados. Isso é obtido por meio de códigos de confirmação que permitem ao eleitor conferir que seu voto está incluso no resultado, sem oferecer outras informações sobre o voto.

É uma evolução do sistema Punchscan, desenvolvido pela mesma equipe junto com outros contribuidores.

Uma versão aprimorada, chamada Scantegrity II, desenvolvida pelos mesmos autores, utiliza tinta invisível para obter melhorias na usabilidade e na resolução de disputas (CHAUM; CARBACK

et al., 2008). É possível verificar o resultado de uma eleição de forma independente de software pelo seu uso de técnicas criptográficas.

O eleitor tem a confirmação da contagem do seu voto por meio de códigos revelados com uma tinta invisível no preenchimento da cédula. A aleatoriedade destes códigos também provê resistência à coerção e garante o segredo do voto mesmo que sejam revelados.

Um teste público foi realizado em 2009 na cidade estadunidense de Takoma Park, em Maryland, sob a hipótese de que o Scantegrity seria um sistema confiável, fácil de usar e administrar, seria aceito como alternativa de alta confiança e que os eleitores auditariam seus votos posteriormente (SHERMAN et al., 2010).

O teste revelou que o Scantegrity tem um tempo médio por eleitor de 167 segundos, o dobro do tempo médio das eleições gerais do Brasil em urnas eletrônicas, que varia entre 61 e 85 segundos (TSE, 2014). Apesar de considerarem que o sistema foi fácil de usar, muitos eleitores não usaram corretamente as ferramentas de privacidade fornecidas e não entenderam que poderiam auditar seu voto online.

Além dos eleitores, os mesários também responderam ter dificuldades com o uso do sistema, alegando que havia muita informação e questionando se as melhorias do Scantegrity valiam a pena pela dificuldade adicional.

2.2.3 Prêt à Voter

O sistema Prêt à Voter, descrito em Ryan, Bismark et al. (2009), é um sistema que procura garantir a precisão da contagem e a privacidade do voto de forma independente do software e hardware utilizado no processo. Em particular, este sistema permite que eleitores confirmem seu voto no resultado final, e previne a coerção

e compra de votos.

No Prêt à Voter, a lista de candidatos na cédula é ordenada de acordo com uma variável criptográfica, como no Punchscan. Isso garante o segredo do voto no recibo e mitiga qualquer tendência a favor do candidato no topo da lista como pode ocorrer em ordens fixas.

É um sistema com auditabilidade fim-a-fim, e os eleitores podem conferir que seu voto foi contabilizado corretamente nos resultados públicos da votação.

Diversas modificações foram feitas ao sistema para adicionar novas características, como métodos de verificação simplificados para eleitores leigos (LUNDIN; RYAN, 2008), aprimoramentos no sistema criptográfico (RYAN; SCHNEIDER, 2006) e um híbrido entre Punchscan e Prêt à Voter elaborado para eleições por correios (POPOVENIUC; LUNDIN, 2007).

3 TRABALHOS CORRELATOS

Alguns sistemas de eleição remotos com funções criptográficas estão presentes na literatura e disponíveis para consulta. Consideramos apenas os sistemas livres e de código aberto, descartando a análise de soluções comerciais pela impossibilidade de verificar e utilizar seus conceitos. Notadamente, descrevemos brevemente os sistemas ADDER, Civitas e Helios.

3.1 ADDER

O sistema ADDER, descrito em Kiayias, Korman e Walluck (2006), é baseado em criptografia homomórfica e chaves distribuídas entre as partes interessadas, implementado utilizando um servidor de registro (*bulletin board*), um servidor de autenticação (*gatekeeper*) separado do *bulletin board* e um cliente *web*.

Votações utilizando o sistema ADDER são transparentes, de forma que todos os dados no *bulletin board* são acessíveis publicamente. Estes dados incluem os votos criptografados, as chaves públicas e a contagem final dos votos. O *bulletin board* não contém dados secretos. As votações também possuem verificabilidade universal, isto é, qualquer interessado pode auditar o procedimento inspecionando a transcrição da eleição.

Estas características são importantes para sistemas criptográficos, uma vez que o mínimo possível de informações secretas é desejável para reduzir as possibilidades de falha do sistema, sendo uma proteção contra a segurança por obscuridade.

O ADDER utiliza chaves criptográficas distribuídas entre múltiplas autoridades responsáveis, que podem ser os concorrentes

da eleição e responsáveis pelo processo eleitoral, e utiliza um limiar para assegurar que o resultado final só pode ser revelado mediante cooperação de uma dada quantidade de autoridades. Qualquer tentativa de interferir no processo requer a corrupção de uma grande quantidade de autoridades, portanto, os eleitores podem garantir confiabilidade no processo e até participar ativamente dele.

3.2 CIVITAS

O sistema Civitas, descrito em Clarkson, Chong e Myers (2008), tem suas raízes no *Condorcet Internet Voting System* (CIVS), um sistema de pesquisa online e aberto que permite que qualquer usuário acesse o sistema e crie uma eleição.

O Civitas é projetado para resistir a coerção, permitindo que um eleitor ameaçado utilize credenciais falsas para votar sem modificação aparente no sistema, que será descartado na apuração, baseado em protocolo descrito por Juels, Catalano e Jakobsson (2005).

O eleitor também não pode se corromper propositalmente, pois não há prova de voto mesmo em caso de interação com o agressor no momento do voto, pois permite votos múltiplos considerando apenas o voto final no resultado.

O sistema é compatível com diferentes modelos de cédulas, como cédulas de voto em candidato único, voto em conjunto de candidatos, e voto por classificação (opções ordenadas por preferência).

O Civitas utiliza um sistema criptográfico distribuído com resistência a falhas. Neste caso, uma falha acontece quando um agente responsável (autoridade) pela eleição desvia do protocolo padrão, seja por comportamento malicioso, erros não intencionais ou simplesmente pela não colaboração (DAVIS; CHMELEV; CLARK-

SON, 2008). Para estas situações, há uma quantidade mínima configurável de agentes necessários para realizar a apuração menor ou igual à quantidade total de agentes responsáveis, de forma semelhante ao ADDER.

3.3 HELIOS

O sistema Helios, descrito em Adida (2008), utiliza um protocolo que possui similaridades com ADDER. Foi desenvolvido como uma plataforma de eleição online, e suporta integração com diversos sistemas de *login*, como CAS e OAuth.

A construção da cédula é baseada no protocolo descrito por Josh Benaloh (2006), que separa os processos de preparação e envio da cédula, de forma que mesmo um participante que não seja apto para votar possa acompanhar o processo de preparação, e apenas no passo final da cabine de votação a autenticação do eleitor se faz necessária.

De um ponto de vista técnico, a descentralização da verificação do Helios permite alta confiabilidade no processo, uma vez que não há um ponto central de falha na votação e cada participante pode confirmar a contabilização do seu voto.

No entanto, o sistema sofre com problemas de usabilidade (KARAYUMAK et al., 2011), já que os eleitores não possuem prática suficiente para tal procedimento, muitas vezes ignorando a auditoria, que é uma das vantagens de sistemas auditáveis fim-a-fim.

4 IMPLEMENTAÇÃO

Na elaboração de um novo protocolo e sistema de eleição, alguns desafios devem ser superados ou considerados para garantir a integridade dos resultados. Adotando a mentalidade de segurança, rapidamente encontraríamos os problemas dos sistemas mais primitivos citados anteriormente, como a falta de privacidade da votação por aclamação e as possibilidades de adulteração em urnas eletrônicas.

Neste trabalho, foi desenvolvida uma especificação de um sistema remoto de eleição, composto de um protocolo de procedimento de eleição baseado no ADDER, da descrição da interação com serviços terceirizados de autenticação de usuários e de uma biblioteca de primitivas criptográficas, que foi desenvolvida para uso neste sistema.

4.1 CADASTRO E AUTENTICAÇÃO DE ELEITORES

O subsistema de cadastro e autenticação de eleitores é responsável por registrar os dados e confirmar a identidade dos eleitores que acessem o sistema de votação. Nos sistemas citados no Capítulo 3, são utilizados diferentes sistemas de registro e autenticação, notadamente o uso de *tokens* enviados para os eleitores antes da votação.

O Brasil já possui alguns sistemas que utilizam autenticação eletrônica, na forma de certificados digitais, como o e-CNPJ. Esse tipo de autenticação provê muito mais segurança do que assinaturas físicas, pois anula a possibilidade de falsificação de identidade, ou seja, o portador de um certificado digital e-CNPJ é o seu dono

dada a dificuldade em se falsificar um certificado.

Essa modalidade de autenticação é bastante flexível, sendo utilizada também por bancos para autenticar os dispositivos utilizados pelos clientes, sejam celulares, tablets e similares, sem a necessidade de senhas ou *tokens*, mas é possível combinar com estes para prover ainda mais segurança. Idealmente, deve-se utilizar mais de um fator de autenticação para evitar clonagem ou roubo das credenciais, sendo bastante comum na internet os *tokens* baseados no tempo (*time-based one-time password* ou **TOTP**).

O sistema Helios, de acordo com o código em seu repositório, tem suporte a *plugins* de autenticação, suportando padrões abertos de *single sign-on* como CAS, OAuth, Shibboleth e sistemas de identificação federados como o OpenID Connect através destes. Pode também, teoricamente, suportar autenticação através de *smart cards* e outras formas de certificados digitais, mas seu repositório não integra nenhuma destas soluções de autenticação citadas atualmente.

A terceirização do serviço de autenticação permite delegar a confiabilidade deste para outra autoridade dedicada. Por exemplo, diversas universidades possuem serviços de autenticação centralizada utilizando algum dos protocolos citados anteriormente, e a própria entidade garante a autenticidade dos seus registros e a autorização do seu uso. Isso reduz a superfície de ataque do sistema de eleição e o torna mais flexível e adaptável a novas tecnologias.

Para a implementação de referência desenvolvida neste trabalho, optou-se por utilizar o protocolo OpenID Connect (SAKIMURA et al., 2014), utilizando como servidor de autorização o Google. Desta forma, a autenticação de usuários é realizada por um terceiro, e o servidor do registro de votos não gerencia os usuários, se limitando a armazenar apenas o nome e e-mail.

Utiliza-se também a tecnologia *Credential Management*

API (**CM-API**) para armazenar e recuperar credenciais federadas diretamente no *browser*, simplificando o fluxo de reautenticação de uma sessão expirada (WEST, 2018). Essa tecnologia ainda é considerada experimental, estando disponível somente no navegador Chrome no momento de escrita deste trabalho.

A combinação destas duas tecnologias permite armazenar o mínimo possível de informação sobre o eleitor no servidor de eleição, delegando a tarefa de autenticação para outros serviços de confiança do próprio usuário, como o navegador de sua preferência e o serviço de autenticação a ser consultado.

A **CM-API** também prevê suporte para autenticação por senha e por chave pública. Para a autenticação por chave pública, é possível utilizar dispositivos de chave assimétrica, como *smart cards* contendo certificados digitais para autenticação física dos usuários, se este nível de segurança for desejado. Nenhum navegador suporta este método no momento da escrita deste trabalho, mas Chrome e Firefox já possuem implementações parciais.

4.2 REGISTRO PÚBLICO DE VOTOS

O registro de votos é o subsistema responsável por receber e armazenar com segurança os votos dos eleitores. Este sistema não deve ter capacidade de abrir os votos depositados, não tendo acesso à chave privada utilizada para decifrá-los. Dessa forma, sua responsabilidade é evitar votos duplicados, permitindo que um eleitor sobrescreva seu voto, e reportar para o sistema de contagem o resultado final.

O subsistema de registro não deve ter acesso à chave privada de nenhuma eleição, para garantir o sigilo do voto. Deve somente fornecer uma interface para a inserção e conferência dos votos, como faz uma urna, repassando o somatório cifrado às auto-

ridades de contagem ao final do processo.

Pode ser implementado como uma camada fina de software sobre um banco de dados que confere a autenticação do usuário com o subsistema de autenticação, armazena os votos válidos de forma cifrada neste banco de dados e provê uma forma de visualização dos votos já inseridos, sem informações sobre os eleitores.

4.3 BIBLIOTECA DE PRIMITIVAS CRIPTOGRÁFICAS

Como esta implementação utiliza rotinas criptográficas genéricas, é possível criar uma biblioteca dedicada a estas primitivas, para ser usada não apenas nesta implementação, mas em qualquer *software* que deseje utilizá-las. Dessa forma, o código pode ser testado mais vigorosamente por aplicações de diferentes finalidades, aumentando a confiabilidade e a probabilidade de que uma falha seja detectada (HÜRSCH; LOPES, 1995).

Esta biblioteca é responsável pela geração dos números primos necessários no criptosistema utilizado, utilidades de verificação de primalidade, geração de pares de chaves e compartilhamento de segredos, que serão descritos nas próximas subseções.

A biblioteca foi denominada **swartz** em homenagem a Aaron Swartz (1986-2013), um programador e ativista que foi preso ao ser acusado de roubar documentos de um serviço de periódicos online para distribuir ilegalmente, acusação esta que nunca foi provada (SIMS, 2011).

4.3.1 Criptosistema de Paillier

Para realizar a criptografia dos votos para armazenamento, é necessário um sistema criptográfico de chaves assimétricas em que o subsistema de registro de votos possa realizar a contagem

sem conhecimento da chave privada para descriptografar os dados. Isso implica que o sistema criptográfico utilizado tenha propriedades homomórficas, para que seja possível realizar operações com resultado conhecido diretamente no texto cifrado.

A propriedade homomórfica é utilizada na contagem para que o resultado criptografado possa ser publicado para então ser aberto pelas autoridades detentoras da chave privada. Deve ser possível gerar o somatório dos votos apenas com a cifragem de cada voto.

O criptosistema de Paillier (PAILLIER, 1999) foi escolhido por possuir estas características, sendo um criptosistema de chaves assimétricas com propriedades homomórficas cuja complexidade é dada pela hipótese de intratabilidade da residuosidade composta de decisão (do inglês *decisional composite residuosity*, **DCR**).

Embora o ADDER originalmente utilize o criptosistema de ElGamal, durante este trabalho o autor considerou o criptosistema de Paillier mais simples e com pesquisas mais aprofundadas sobre sua segurança no âmbito de eleições online. É plenamente possível que posteriormente seja implementado o criptosistema de ElGamal como alternativa.

Além disso, o criptosistema de Paillier utiliza uma chave efêmera aleatória para cifragem, o que gera uma mensagem cifrada diferente cada vez que a mesma mensagem original é cifrada. Portanto, mesmo se dois eleitores votarem exatamente nas mesmas opções, o voto cifrado armazenado no banco de dados será diferente para cada um.

Essa propriedade permite que os votos de um eleitor sejam representados pelos valores 0 e 1 caso o eleitor vote contra ou a favor de uma opção, respectivamente. Se a mensagem cifrada fosse igual para todos os valores iguais, seria fácil observar o registro público da seção 4.2 dos votos e obter informações sobre os votos sem

conhecer a chave privada da eleição.

A chave efêmera não é armazenada e não é necessária para decifrar a mensagem, portanto não é possível reproduzir o voto somente com as informações armazenadas no registro a fim de quebrar o sigilo do voto.

4.3.2 Compartilhamento de segredos de Shamir

Para dividir a responsabilidade de revelar a contagem final de votos entre múltiplas autoridades, se faz necessário um esquema de compartilhamento de segredos. Em Shamir (1979) é descrito um esquema de compartilhamentos baseado em polinômios, cuja ideia central é de que são necessários k pontos para se definir um polinômio de grau $k-1$.

Utilizando este esquema, definimos um polinômio de grau $k-1$, onde k é a quantidade mínima de autoridades necessárias para se revelar os textos cifrados, dentre um total de n autoridades tal que $0 < k \leq n$. Cada autoridade calcula sua decifragem parcial c_i que será utilizada para reconstruir a mensagem original m quando combinadas ao menos k decifragens parciais.

A definição do criptosistema de Paillier modificado para suportar o compartilhamento de segredos de Shamir utilizado neste trabalho encontra-se em Fouque, Poupard e Stern (2000). Embora haja uma implementação mais genérica em Damgård, Jurik e Nielsen (2010), esta possui restrições quanto aos valores de k em relação a n que foram consideradas indesejáveis.

4.4 PROTOCOLO DE UMA ELEIÇÃO

O protocolo de uma eleição utilizado no ADDER, e portanto no qual é inspirado este trabalho, se concentra em torno do

registro digital público de votos, chamado de *bulletin board* ou *ledger*, e está descrito em Josh Daniel Cohen Benaloh (1987). Este registro consiste em um mural público onde todos os usuários autenticados, tanto eleitores, autoridades e administradores podem inserir dados.

Na implementação, o *ledger* consiste em um banco de dados relacional acessado através de uma interface *web* que armazena as credenciais de uma eleição e versões criptografadas dos votos, e um servidor de autenticação que emite *tokens* com os quais os usuários podem se autenticar com o *ledger* e realizar ações de acordo com sua permissão. Uma interface *web* acessa os dados destes servidores de forma transparente.

O procedimento de execução de uma eleição é dividido em passos sequenciais para criar as informações necessárias, receber os votos e calcular o resultado final, listados:

1. Enviar os parâmetros da nova eleição
2. Envio das chaves privadas das autoridades
3. Geração e distribuição do par de chaves
4. Coleta de votos
5. Totalização dos votos cifrados
6. Decifragens parciais pelas autoridades
7. Combinação e publicação dos resultados

Para criar uma nova eleição, um administrador do sistema envia os parâmetros da nova eleição para o *ledger* através da interface *web*. Estes parâmetros são os identificadores da eleição, a lista de n autoridades, a lista de eleitores, a quantidade mínima m_{min}

e máxima m_{max} de candidatos por voto, a lista de candidatos, o limite mínimo de autoridades t e os horários limites para os passos seguintes. Estas informações ficam publicamente disponíveis no *ledger* para conferência.

No segundo passo, as autoridades devem submeter suas chaves públicas para o sistema, para que toda a comunicação de informação sensível entre o sistema e a autoridade se dê por forma criptografada, nunca em texto claro. A autoritade também deve submeter um desafio para provar que detém a chave privada relacionada à chave pública enviada. Se menos de t autoridades enviarem chaves públicas até o prazo expirar para este passo, a eleição é encerrada com falha.

No terceiro passo, o administrador que criou a eleição solicita que o sistema gere o par de chaves da eleição. Para isso, utiliza o algoritmo de geração de chaves descrito em Fouque, Poupard e Stern (2000), que cria uma chave pública, divide a chave privada entre as autoridades que participaram do passo anterior e envia cada parte por e-mail de forma criptografada. A chave pública e as chaves de verificação são publicados no *ledger* junto com os outros detalhes da eleição gerados até agora. Se menos de t autoridades confirmarem o recebimento de suas chaves privadas parciais até o prazo expirar para este passo, a eleição é encerrada com falha.

Antes do quarto passo, que é a votação de fato, o administrador deve publicar a lista de eleitores no *ledger*. Os eleitores são identificados unicamente por e-mail, e como especificado anteriormente, são autenticados por um servidor de autenticação externo.

Durante a votação, somente os eleitores submetidos anteriormente podem inserir seus votos no *ledger*. Os votos são cifrados e acompanhados de uma prova de formação correta, de forma que o eleitor pode conferir se seu voto cifrado está corretamente publicado e o sistema pode garantir que o eleitor votou em pelo menos

m_{min} opções e em até m_{max} opções. O eleitor pode substituir seu voto quantas vezes desejar, de forma a mitigar tentativas de coerção, mas não impedindo-as.

Este protocolo explicitamente não lida com o problema da coerção, pois com a natureza descentralizada do sistema proposto é excessivamente complexo controlar cada terminal de votação. Em vez disso, a possibilidade de substituir o voto a qualquer momento permite que o eleitor vote até mesmo em frente a um agressor e posteriormente desfaça seu voto. Esta é a mesma abordagem do sistema Helios.

Encerrada a coleta de votos, no quinto passo o sistema faz a totalização dos votos, somando-os de forma homomórfica baseada nas propriedades do criptosistema de Paillier. O sistema deve considerar somente os votos corretamente formados e mais recentes de cada eleitor.

No sexto passo, as autoridades submetem suas decifragens parciais do somatório dos votos, obtidos utilizando as chaves privadas parciais distribuídas no terceiro passo, acompanhados de provas da decifragem correta. Pelo esquema de Fouque, Poupard e Stern (2000), nenhuma autoridade sozinha pode obter o resultado final da eleição, somente uma combinação de pelo menos t decifragens parciais permite encontrar o resultado total decifrado. Se menos de t autoridades enviarem resultados parciais válidos até o prazo expirar para este passo, a eleição é encerrada com falha.

No sétimo e último passo, o sistema combina as decifragens parciais e gera a totalização, que é então publicada no *ledger* junto com os outros dados da eleição, e encerra com sucesso.

O protocolo apresentado é uma simplificação do protocolo ADDER, uma vez que a geração de chaves do criptosistema ElGamal utilizado no ADDER é dividido em mais passos. A ideia de simplificar o protocolo e reduzir a necessidade de interação tam-

bém tem o objetivo de facilitar o uso por autoridades e eleitores com menor conhecimento técnico.

4.5 MODULARIDADE

Seguindo tendências modernas para o desenvolvimento de sistemas *web*, o *software* que faz parte desta implementação é logicamente dividido entre *back-end*, composto pelo *ledger* e os serviços que este acessa (como os serviços de autenticação), e o *front-end*, composto pelos clientes responsáveis pela interação com o usuário. Desta forma, é possível que cada parte tenha menos responsabilidade e possibilidades de falha do que o tradicional método, onde uma única aplicação faz tanto a interação com o usuário quanto a manipulação do banco de dados (LANTHALER; GÜTL, 2012).

Outra vantagem desta abordagem é que desenvolvedores independentes podem construir outros clientes, como aplicativos para celular ou até mesmo terminais especializados para esquemas de eleição alternativos baseados nesta implementação, sem a necessidade de modificar o *back-end* para suportar tais adaptações.

Esta tendência vem em conjunto com *frameworks* para interfaces de usuário como Angular, React e Vue, que transferem a responsabilidade de apresentar as informações para o cliente, em vez de renderizar as páginas HTML no servidor. Técnicas agressivas de *cache* também favorecem, já que a quantidade de informação transmitida entre o cliente e o servidor se limita ao mínimo necessário para cada atividade e há maior reutilização de código quando o conteúdo é construído dinamicamente (SOUDERS, 2008).

4.5.1 Comunicação entre os módulos

A comunicação entre o cliente e o servidor se dá através de uma linguagem de representação de dados, sendo a mais comum atualmente o **JSON**. O canal de comunicação precisa ser seguro e privado, portanto sempre ocorrendo por um canal com protocolo SSL ou TLS, como o HTTPS (RESCORLA, 2000) no caso de um cliente de *browser*.

Computadores e celulares são rotineiramente atacados por software malicioso, como vírus e *malwares*, e requerem cuidado especial da parte do usuário, no caso o eleitor. A segurança neste lado não pode ser garantida pelo sistema eleitoral e é, portanto, uma preocupação para a adoção de sistemas de eleição remotos, mas novas tecnologias e padrões recentemente desenvolvidos e implementados pelos navegadores permitem maiores níveis de segurança para aplicações *web*.

4.5.1.1 Mitigação de vulnerabilidades

Dentre as tecnologias desenvolvidas nos últimos anos para mitigar vulnerabilidades no *browser*, são as mais conhecidas: *Content Security Policy* (**CSP**), que visa controlar recursos que uma aplicações *web* pode executar (WEST, 2016); *HTTP Strict Transport Security* (**HSTS**), que protege de ataques de *downgrade* de protocolo, forçando acesso somente por canais criptografados e seguros (HODGES; JACKSON; BARTH, 2012); *HTTP Public Key Pinning* (**HPKP**), que associa chaves públicas com uma aplicação e evita riscos de ataques com certificados forjados (EVANS; PALMER; SLEEVI, 2015); *Subresource Integrity* (**SRI**), que provê formas do navegador assegurar que recursos foram corretamente transferidos sem manipulação (AKHAWE et al., 2016); e as inúmeras recomendações da iniciativa *Open Web Application Security*

Project (**OWASP**), uma organização sem fins lucrativos que distribui documentação e ferramentas para segurança em aplicações *web*. A maioria destas soluções diz respeito ao cliente, não ao servidor.

No servidor, além dos cuidados com infecção maliciosa, que poderiam ser evitadas por profissionais técnicos, também é necessário lidar com outros tipos de invasão e negação de serviço. Um ou mais *insiders*, sejam eles técnicos ou programadores responsáveis pelo sistema podem manipular o software de forma indetectável, e as recentes acusações dos Estados Unidos sobre uma suposta manipulação nas suas eleições por espiões da Rússia certamente não melhora a opinião pública (BADAWY; FERRARA; LERMAN, 2018).

Ataques de negação de serviço (**DoS**, na sigla para *Denial of Service*) ocorrem quando um agressor acessa um serviço com uma grande quantidade de dispositivos simultaneamente, com o objetivo de congestionar e sobrecarregar a rede e o servidor, impedindo que outros usuários o acessem. Ataques desse tipo em larga escala tipicamente utilizam milhares ou milhões de computadores previamente infectados com vírus, dificultando sua mitigação, pois não há uma origem consistente dos acessos. Este tipo de ataque também pode ser utilizado para distrair equipes de segurança e explorar outras falhas no sistema enquanto as defesas estão focadas em mitigar a sobrecarga.

Uma forma de ataque de negação de serviço conhecida como *Slowloris*, que consiste em abrir uma quantidade de conexões com o servidor maior do que ele pode suportar, mantendo-as abertas e impedindo que o servidor aceite novas conexões de usuários legítimos, foi utilizada em sites do governo iraniano durante as eleições presidenciais de 2009 após inúmeros protestos (ZDRNJA, 2009).

5 CONSIDERAÇÕES FINAIS

Neste trabalho, foi desenvolvida uma implementação de sistema de eleição online utilizando um esquema de criptografia homomórfica, com compartilhamento da chave privada entre as autoridades da eleição, e cuja interface com o usuário é desacoplada do sistema de armazenamento de dados, a fim de reduzir a quantidade de informação trocada e, com isso, diminuir a superfície de ataques.

Este sistema é fundamentado em um protocolo robusto, porém simplificado para comportar eleições gerenciadas por autoridades e utilizadas por eleitores com menor conhecimento técnico sobre as primitivas criptográficas utilizadas.

Embora devamos tomar cuidado ao desenvolvermos uma alternativa remota para um novo sistema eleitoral, é bastante satisfatório observar a evolução da segurança computacional no passado recente. Enquanto eleições tem sido aplicadas durante muitos séculos, e computadores só tenham sido inventados num passado relativamente recente, eles tem rapidamente se tornado potenciais alternativas para os meios de diversas atividades, entre elas as eleições.

O uso de sistemas criptográficos pode reforçar em ordens de magnitude a segurança e privacidade das eleições, de forma muito mais eficiente e com menos interferência humana do que os sistemas rudimentares de cédulas de papel, por exemplo. Isso certamente tornará eleições mais íntegras e protegidas quando estas premissas forem devidamente aplicadas.

As ferramentas necessárias para o desenvolvimento de sis-

temas eleitorais remotos estão em pleno desenvolvimento, como demonstrado nas novas tecnologias de segurança para navegadores e sistemas criptográficos modernos previamente citados. Estas ferramentas permitirão novos níveis de segurança, mais próximos do necessário para eleições.

Analisando o histórico da humanidade e da democracia, podemos esperar que no futuro as eleições serão realizadas de forma descentralizada, e um sistema como o proposto forma uma sólida base sobre a qual as eleições em qualquer instância podem se tornar mais seguras, íntegras e práticas, considerando a intensa penetração dos dispositivos móveis e informação de todos os serviços públicos.

O objetivo de construir uma solução com alto nível de segurança e privacidade foi atingido, considerando a utilização do protocolo ADDER como base, e a segurança provida pelo criptosistema de Paillier. Este criptosistema utiliza primitivas criptográficas bem estabelecidas, e sua segurança é baseada na dificuldade da fatoração de inteiros, que também provê a segurança do criptosistema RSA, amplamente utilizado.

É importante salientar que as operações realizadas pela implementação são bastante limitadas, de forma que o sistema final realiza somente a tarefa de gerenciar a eleição, e explicitamente delega a tarefa de autenticação para um terceiro, a fim de minimizar a responsabilidade e melhorar a extensibilidade do sistema.

5.1 TRABALHOS FUTUROS

Um assunto que não foi abordado neste trabalho foi o aspecto legal de uma solução descentralizada como a proposta, ou como categorizar o dispositivo utilizado pelo eleitor em uma eleição descentralizada. Caso estes dispositivos sejam considerados parte do equipamento eleitoral, e portanto sujeitos à lei eleitoral, que cri-

minaliza tentativas de quebra de sigilo e limita propaganda eleitoral, seria praticamente impossível fiscalizar dado que milhares de dispositivos estão em uso, portanto essa mudança de paradigma exigiria mudanças na lei eleitoral caso a implementação seja usada numa eleição governamental.

O uso de uma infraestrutura de chaves públicas (**ICP**) para gerenciamento das chaves das autoridades no segundo passo do protocolo da eleição terceirizaria essa responsabilidade para um diretório dedicado, reduziria ainda mais as responsabilidades do sistema implementado e poderia tornar este passo totalmente automático, caso todas as autoridades já tivessem chaves públicas inseridas em uma ICP. Esta modificação no protocolo também aumenta o grau de confiança na identidade da autoridade, pois a tarefa de submeter a chave pública é feita por um terceiro, e não pela própria autoridade (NASH; DUANE; JOSEPH, 2001).

Seria positivo avaliar formas de remover a participação da entidade do administrador do sistema de uma eleição, tornando o processo da eleição dependente somente das autoridades e eleitores. Esta é mais uma mudança de paradigma, pois eleições historicamente são gerenciadas por uma entidade dedicada, como o Tribunal Superior Eleitoral no caso das eleições governamentais, e demais comissões eleitorais.

Uma tecnologia recente com potencial para melhorar a integridade do processo de uma eleição são *blockchains*, registros de transações criptograficamente ligados em sequência. *Blockchains* são fundamentadas por consenso descentralizado e poderiam ser usadas para garantir que o sistema não esteja modificando os votos de forma maliciosa ou por simples mal-funcionamento (CHRISTIDIS; DEVETSIKIOTIS, 2016). Uma preocupação que deve ser tomada ao se introduzir uma *blockchain* num sistema eleitoral é de prevenir o ordenamento dos votos, de forma a evitar a identificação

do voto de um eleitor específico.

Por fim, a implementação proposta precisa de uma verificação formal de seus aspectos de segurança antes de ser utilizada em uma eleição de grande escala, onde o nível de segurança proposto se faz necessário, como numa eleição governamental ou de uma grande empresa. Embora não tenha sido proposto nenhum novo algoritmo, é importante verificar a coesão de todos os procedimentos utilizados.

REFERÊNCIAS

ADIDA, Ben. Helios: Web-based Open-Audit Voting. In: USENIX security symposium. [S.l.: s.n.], 2008. v. 17, p. 335–348.

AKHAWE, Devdatta et al. **Subresource Integrity**. [S.l.], jun. 2016. http://www.w3.org/TR/2016/REC-SRI-20160623/.

ALVIM, Frederico Franco. Integridade eleitoral: significado e critérios de qualificação. **Ballot**, v. 1, n. 2, p. 213–228, 2015.

ARANHA, Diego F et al. Vulnerabilidades no software da urna eletrônica brasileira, 2012.

BADAWY, Adam; FERRARA, Emilio; LERMAN, Kristina. Analyzing the Digital Traces of Political Manipulation: The 2016 Russian Interference Twitter Campaign. **arXiv preprint arXiv:1802.04291**, 2018.

BENALOH, Josh. Simple Verifiable Elections. **EVT**, v. 6, p. 5–5, 2006.

BENALOH, Josh Daniel Cohen. Verifiable secret-ballot elections. Yale University, 1987.

BENALOH, Josh et al. End-to-end verifiability. **arXiv**, 2015.

BUCHSTEIN, Hubertus. Public voting and political modernization: Different views from the 19th century. In: SCRUTIN secret et vote public, huis clos et débat ouvert, seminar/first draft, Paris: Collège de de France. [S.l.: s.n.], 2010. p. 17–26.

CHAUM, David; CARBACK, Richard et al. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. **EVT**, v. 8, p. 1–13, 2008.

CHAUM, David; ESSEX, Aleks et al. Scantegrity: End-to-end voter-verifiable optical-scan voting. **IEEE Security & Privacy**, IEEE, v. 6, n. 3, 2008.

CHRISTIDIS, Konstantinos; DEVETSIKIOTIS, Michael. Blockchains and smart contracts for the internet of things. **Ieee Access**, IEEE, v. 4, p. 2292–2303, 2016.

CLARKSON, Michael R; CHONG, Stephen; MYERS, Andrew C. Civitas: Toward a secure voting system. In: IEEE. SECURITY and Privacy, 2008. SP 2008. IEEE Symposium on. [S.l.: s.n.], 2008. p. 354–368.

CONEY, Lillie et al. Towards a privacy measurement criterion for voting systems. In: DIGITAL GOVERNMENT SOCIETY OF NORTH AMERICA. PROCEEDINGS of the 2005 national conference on Digital government research. [S.l.: s.n.], 2005. p. 287–288.

CRANOR, Lorrie Faith. In search of the perfect voting technology: No easy answers. In: SECURE Electronic Voting. [S.l.]: Springer, 2003. p. 17–30.

DAMGÅRD, Ivan; JURIK, Mads; NIELSEN, Jesper Buus. A generalization of Paillier's public-key system with applications to electronic voting. **International Journal of Information Security**, Springer, v. 9, n. 6, p. 371–385, 2010.

DAVIS III, John M; THOMAS, Shelby. **Direct recording electronic voting machine and voting process**. [S.l.]: Google Patents, dez. 1996. US Patent 5,583,329.

DAVIS, Adam M; CHMELEV, Dmitri;
CLARKSON, Michael R. Civitas: Implementation of a threshold cryptosystem, 2008.

DELAUNE, Stephanie; KREMER, Steve; RYAN, Mark. Coercion-resistance and receipt-freeness in electronic voting. In: IEEE. COMPUTER Security Foundations Workshop, 2006. 19th IEEE. [S.l.: s.n.], 2006. 12–pp.

EVANS, Chris; PALMER, Chris; SLEEVI, Ryan. **Public Key Pinning Extension for HTTP**. [S.l.]: RFC Editor, abr. 2015. 28 p. RFC 7469. (Request for Comments, 7469). DOI: 10.17487/RFC7469. Disponível em: <https://rfc-editor.org/rfc/rfc7469.txt>.

EVERETT, Sarah P; BYRNE, Michael D;
GREENE, Kristen K. Measuring the usability of paper ballots: Efficiency, effectiveness, and satisfaction. In: SAGE PUBLICATIONS SAGE CA: LOS ANGELES, CA, 24. PROCEEDINGS of the Human Factors and Ergonomics Society Annual Meeting. [S.l.: s.n.], 2006. v. 50, p. 2547–2551.

FELDMAN, Ariel J; HALDERMAN, J Alex;
FELTEN, Edward W. Security analysis of the Diebold AccuVote-TS voting machine. Center for Information Technology Policy, Princeton University, 2006.

FISHER, Kevin; CARBACK, Richard; SHERMAN, Alan T. Punchscan: Introduction and system definition of a high-integrity election system. In: PROCEEDINGS of Workshop on Trustworthy Elections. [S.l.: s.n.], 2006.

FOUQUE, Pierre-Alain; POUPARD, Guillaume;
STERN, Jacques. Sharing decryption in the context of voting

or lotteries. In: SPRINGER. INTERNATIONAL Conference on Financial Cryptography. [S.l.: s.n.], 2000. p. 90–104.

GILLESPIE, A.J. **Voting-machine.** [S.l.]: Google Patents, jul. 1899. US Patent 628,905. Disponível em: <https://www.google.com/patents/US628905>.

HODGES, Jeff; JACKSON, Collin; BARTH, Adam. **HTTP Strict Transport Security (HSTS)**. [S.l.]: RFC Editor, nov. 2012. 46 p. RFC 6797. (Request for Comments, 6797). DOI: 10.17487/RFC6797. Disponível em: <https://rfc-editor.org/rfc/rfc6797.txt>.

HÜRSCH, Walter L; LOPES, Cristina Videira. Separation of concerns. Citeseer, 1995.

JONES, D.; SIMONS, B. **Broken Ballots: Will Your Vote Count?** [S.l.]: CSLI Publications, 2012. (CSLI lecture notes). ISBN 9781575866376. Disponível em: <https://books.google.com.br/books?id=Y5TgygAACAAJ>.

JUELS, Ari; CATALANO, Dario; JAKOBSSON, Markus. Coercion-resistant electronic elections. In: ACM. PROCEEDINGS of the 2005 ACM workshop on Privacy in the electronic society. [S.l.: s.n.], 2005. p. 61–70.

KARAYUMAK, Fatih et al. Usability Analysis of Helios-An Open Source Verifiable Remote Electronic Voting System. **EVT/WOTE**, v. 11, n. 5, 2011.

KIAYIAS, Aggelos; KORMAN, Michael; WALLUCK, David. An internet voting system supporting user privacy. In: IEEE. COMPUTER Security Applications Conference, 2006. ACSAC'06. 22nd Annual. [S.l.: s.n.], 2006. p. 165–174.

KOHNO, Tadayoshi et al. Analysis of an electronic voting system. In: IEEE. SECURITY and Privacy, 2004. Proceedings. 2004 IEEE Symposium on. [S.l.: s.n.], 2004. p. 27–40.

LANTHALER, Markus; GÜTL, Christian. On using JSON-LD to create evolvable RESTful services. In: ACM. PROCEEDINGS of the Third International Workshop on RESTful Design. [S.l.: s.n.], 2012. p. 25–32.

LEIB, Jonathan I; DITTMER, Jason. Florida's residual votes, voting technology, and the 2000 election. **Political geography**, Elsevier, v. 21, n. 1, p. 91–98, 2002.

LESLIE, Frank. Stuffer's Ballot-Box. **Frank Leslie's Illustrated Newspaper**, p. 92, jul. 1856.

LUNDIN, David; RYAN, Peter. Human readable paper verification of Pret a Voter. **Computer Security-ESORICS 2008**, Springer, p. 379–395, 2008.

NASH, Andrew; DUANE, William; JOSEPH, Celia. **PKI: Implementing and Managing E-security**. [S.l.]: McGraw-Hill, Inc., 2001.

NICOLAU, J. **Eleições no Brasil: Do Império aos dias atuais**. [S.l.]: Zahar, 2012. (Nova Biblioteca de Ciências Sociais). ISBN 9788537809105. Disponível em: <https://books.google.com.br/books?id=1Aeipw-88HMC>.

PAILLIER, Pascal. Public-key cryptosystems based on composite degree residuosity classes. In: SPRINGER. INTERNATIONAL Conference on the Theory and Applications of Cryptographic Techniques. [S.l.: s.n.], 1999. p. 223–238.

POPOVENIUC, Stefan; LUNDIN, David. A simple technique for safely using punchscan and pret a voter in mail-in elections. **E-Voting and Identity**, Springer, p. 150–155, 2007.

RESCORLA, Eric. **HTTP Over TLS**. [S.l.]: RFC Editor, mai. 2000. 7 p. RFC 2818. (Request for Comments, 2818). DOI: 10.17487/RFC2818. Disponível em: <https://rfc-editor.org/rfc/rfc2818.txt>.

RIVEST, Lecturer Ron; LEDLIE, Scribe et al. Lecture notes 15: Voting, homomorphic encryption. Citeseer, 2002.

ROTH, Susan King. Disenfranchised by design: voting systems and the election process. **Information Design Journal**, John Benjamins Publishing Company, v. 9, n. 1, p. 29–38, 1998.

RYAN, Peter YA; BISMARK, David et al. Prêt à voter: a voter-verifiable voting system. **IEEE transactions on information forensics and security**, IEEE, v. 4, n. 4, p. 662–673, 2009.

RYAN, Peter; SCHNEIDER, Steve. Prêt à voter with re-encryption mixes. **Computer Security–ESORICS 2006**, Springer, p. 313–326, 2006.

SAKIMURA, Nat et al. OpenID Connect Core 1.0 incorporating errata set 1. **The OpenID Foundation, specification**, 2014.

SCHEFFEL, Glauco Vinicius. Segurança na avaliação de conhecimento em contexto não presencial. Florianópolis, SC, 2002.

SCHNEIER, Bruce. Inside the twisted mind of the security professional. **Wired Magazine**, v. 16, n. 3, 2008.

SHAMIR, Adi. How to Share a Secret. **Commun. ACM**, ACM, New York, NY, USA, v. 22, n. 11, p. 612–613, nov. 1979. ISSN 0001-0782. DOI: 10.1145/359168.359176. Disponível em: <http://doi.acm.org/10.1145/359168.359176>.

SHERMAN, Alan T et al. Scantegrity Mock Election at Takoma Park. In: ELECTRONIC Voting. [S.l.: s.n.], 2010. p. 45–61.

SIMS, Nancy. Library licensing and criminal law: The Aaron Swartz case. **College & Research Libraries News**, v. 72, n. 9, p. 534–537, 2011.

SOUDERS, Steve. High-performance web sites. **Communications of the ACM**, ACM, v. 51, n. 12, p. 36–41, 2008.

SPRINGALL, Drew et al. Security Analysis of the Estonian Internet Voting System. In: PROCEEDINGS of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale, Arizona, USA: ACM, 2014. (CCS '14), p. 703–715. ISBN 978-1-4503-2957-6. DOI: 10.1145/2660267.2660315. Disponível em: <http://doi.acm.org/10.1145/2660267.2660315>.

TSE. **Confira a estimativa de tempo médio de votação no segundo turno das Eleições 2014**. [S.l.: s.n.], out. 2014. http://www.tse.jus.br/imprensa/noticias-tse/2014/Outubro/confira-a-estimativa-de-tempo-medio-de-votacao-no-segundo-turno-das-eleicoes-2014. Acesso em: 4 nov 2018.

WEST, Mike. **Content Security Policy Level 3**. [S.l.], set. 2016. https://www.w3.org/TR/2016/WD-CSP3-20160913/.

WEST, Mike. **Credential Management Level 1**. [S.l.: s.n.], jun. 2018. `https://w3c.github.io/webappsec-credential-management/`. Acesso em: 25 set 2018.

WOLCHOK, Scott et al. Security analysis of India's electronic voting machines. In: ACM. PROCEEDINGS of the 17th ACM conference on Computer and communications security. [S.l.: s.n.], 2010. p. 1–14.

WOLFSON, Arthur M. The ballot and other forms of voting in the italian communes. **The American Historical Review**, JSTOR, v. 5, n. 1, p. 1–21, 1899.

ZDRNJA, Bojan. **InfoSec Handlers Diary Blog - Slowloris and Iranian DDoS attacks**. [S.l.: s.n.], jun. 2009. Disponível em: <`https://isc.sans.edu/diary/Slowloris+and+Iranian+DDoS+attacks/6622`>.

APÊNDICE A: BIBLIOTECA SWARTZ

paillier.py

```
1  import functools
2  import math
3  import operator
4  import secrets
5  from dataclasses import dataclass
6  from typing import Dict, List, Set, Tuple
7
8  from swartz.utils import (eval_poly, generate_prime, generate_sa
9                            modular_inverse)
10
11 DEFAULT_KEYSIZE = 2048
12
13
14 @dataclass
15 class PublicKey:
16     n: int
17
18     @property
19     def g(self):
20         return self.n + 1
21
22     def encrypt(self, message: int) -> int:
23         n, g = self.n, self.g
```

```
24          r = choose_multiplicative_inverse(n)
25          mod = n ** 2
26          return (pow(g, message, mod) * pow(r, n, mod)) % mod
27
28
29  @dataclass
30  class PrivateKey:
31      n: int
32      : int
33      μ: int
34
35      def decrypt(self, cipher: int) -> int:
36          n, , μ = self.n, self., self.μ
37          return ((pow(cipher, , n ** 2) - 1) // n * μ) % n
38
39
40  class KeyPair:
41      def __init__(self, public_key: PublicKey, private_key: PrivateKey
42          assert public_key.n == private_key.n
43          self.public_key = public_key
44          self.private_key = private_key
45
46      def decrypt(self, cipher: int) -> int:
47          return self.private_key.decrypt(cipher)
48
49      def encrypt(self, cipher: int) -> int:
50          return self.public_key.encrypt(cipher)
51
52
53  def generate_keypair(length: int = DEFAULT_KEYSIZE) -> KeyPair:
```

```
54      p, q = generate_prime(length // 2), generate_prime(length //
55
56      n = p * q
57      □ = (p - 1) * (q - 1)
58      μ = modular_inverse(□, n)
59      return KeyPair(PublicKey(n), PrivateKey(n, □, μ))
60
61
62  @dataclass
63  class ThresholdPublicKey:
64      g: int
65      n: int
66      v: int
67      □: int
68
69      def encrypt(self, message: int) -> int:
70          n, g = self.n, self.g
71          r = secrets.randbelow(n - 1) + 1
72          mod = n ** 2
73          return (pow(g, message, mod) * pow(r, n, mod)) % mod
74
75
76  def choose_multiplicative_inverse(n):
77      '''
78      >>> p = generate_prime(128)
79      >>> q = generate_prime(128)
80      >>> x = choose_multiplicative_inverse(p * q)
81      >>> math.gcd(x, p * q)
82      1
83      '''
```

```
84      while True:
85          x = secrets.randbelow(n - 1) + 1
86          if math.gcd(x, n) == 1:
87              return x
88
89
90  @dataclass
91  class Share:
92      id: int
93      value: int
94
95      def __hash__(self):
96          return hash((self.id, self.value))
97
98      def __iter__(self):
99          yield self.id
100         yield self.value
101
102
103 def generate_shares(secret, params, n ) -> List[int]:
104     threshold, servers = params.threshold, params.servers
105     poly = [secret, *(secrets.randbelow(n ) for _ in range(threshold -
106     for i in range(1, servers + 1):
107         yield Share(i, eval_poly(poly, i) % n )
108
109
110 RSAModulus = Tuple[int, int, int, int]
111
112
113 def generate_rsa_modulus(length: int) -> RSAModulus:
114     '''
```

```
115         >>> p, p1, q, q1 = generate_rsa_modulus(128)
116         >>> p == 2 * p1 + 1
117         True
118         >>> q == 2 * q1 + 1
119         True
120         >>> (p - 1) * (q - 1) == 4 * p1 * q1
121         True
122         '''
123         p, p1 = generate_safe_prime(length // 2)
124         q, q1 = generate_safe_prime(length // 2)
125         while p1 == q1:
126             q, q1 = generate_safe_prime(length // 2)
127         return p, p1, q, q1
128
129
130     @dataclass
131     class Params:
132         threshold: int
133         servers: int
134
135         @property
136         def Δ(self):
137             return math.factorial(self.servers)
138
139
140     def generate_key(
141             modulus: RSAModulus, params: Params, length: int = DEFAU
142     ) -> Tuple[ThresholdPublicKey, Dict[int, int], Dict[int, int]]:
143         p, p1, q, q1 = modulus
144
145         n = p * q
```

```
146
147     n_squared = n * n
148
149     # fouque
150     [ ] = 4 * p1 * q1
151     print('  =', [ ])
152     assert math.gcd([ ], n) == 1
153
154     [ ] = choose_multiplicative_inverse(n)
155     print('  =', [ ])
156
157     m = p1 * q1
158     secret_key = [ ] * m
159
160     a = choose_multiplicative_inverse(n)
161     b = choose_multiplicative_inverse(n)
162     g = (pow(1 + n, a, n_squared) * pow(b, n, n_squared)) % n_squared
163     assert math.gcd(a, n) == 1
164     assert math.gcd(b, n) == 1
165     assert math.gcd(g, n_squared) == 1
166     print('a =', a, '\nb =', b, '\ng =', g)
167
168
169     while True:
170         v = pow(secrets.randbelow(n), 2)
171         if math.gcd(v, n_squared) == 1:
172             break
173
174     publ = ThresholdPublicKey(g, n, v, (a * secret_key) % n)
175
```

```
176     # secret_key =   *
177     shares = [*generate_shares(secret_key, params, n * m)]
178
179     Δ = math.factorial(params.servers)
180     vks = [(i, pow(v, Δ * s, n_squared)) for i, s in shares]
181
182     return publ, vks, shares
183
184
185 @dataclass
186 class Decryption:
187     id: int
188     value: int
189     proof: int
190
191     def __hash__(self):
192         return hash((self.id, self.value, self.proof))
193
194     def __lt__(self, rhs):
195         return (self.id, self.value) < (rhs.id, rhs.value)
196
197
198 def share_decryption(cipher, share: Share, key, params):
199     n, v = key.n, key.v
200     i, value = share
201
202     n_squared = n * n
203     c = pow(cipher, 2 * math.factorial(params.servers) * value,
204     proof = pow(v, value, n_squared)
205     return Decryption(i, c, proof)
```

```
206
207
208 def combine(shares: Set[Decryption], key, params: Params) -> int:
209     n, [ ] = key.n, key.[ ]
210     n_squared = n * n
211
212     threshold, Δ = params.threshold, params.Δ
213     c = 1
214
215     shares = sorted(shares[:threshold])
216     for i in shares:
217         μ = Δ
218
219         for j in shares:
220             if i is j:
221                 continue
222
223             if i.id == j.id:
224                 raise ValueError('Two shares with same ID')
225
226             μ *= j.id // (j.id - i.id)
227
228         if μ < 0:
229             c = pow(modular_inverse(i.value, n_squared), 2 * -μ, n_sq
230         else:
231             c *= pow(i.value, 2 * μ, n_squared)
232         c %= n_squared
233         print('c =', c)
234
235     return ((c - 1) // n * modular_inverse(4 * Δ * Δ * [ ], n)) % n
```

utils.py

```python
import secrets
from typing import Tuple


def extended_gcd(a: int, b: int) -> Tuple[int, int, int]:
    """Calculates gcd using extended Euclidean algorithm
    >>> extended_gcd(240, 46)
    (2, -9, 47)
    """
    if a == 0:
        return b, 0, 1
    g, x, y = extended_gcd(b % a, a)
    return g, y - (b // a) * x, x


def modular_inverse(a: int, n: int) -> int:
    """Calculates modular inverse of a mod n, i.e x such that ax
    >>> modular_inverse(77, 5)
    3
    >>> modular_inverse(55, 7)
    6
    >>> modular_inverse(5, 10)
    Traceback (most recent call last):
        ...
    ValueError: 5 is not relatively prime to 10
    """
    g, x, _ = extended_gcd(a, n)
    if g != 1:
        raise ValueError(f'{a} is not relatively prime to {n}')
```

```
30      return x % n
31
32
33  def is_probable_prime(n: int, k: int = 100) -> bool:
34      """Tests whether n is a probable prime with Miller-Rabin primalit
35      >>> is_probable_prime(93)
36      False
37      >>> is_probable_prime(104789)
38      True
39      >>> is_probable_prime(15485943)
40      False
41      >>> is_probable_prime(32416190071)
42      True
43      """
44      if n == 2:  # pragma: no cover
45          return True
46
47      d, r = n - 1, 0
48      while d % 2 == 0:
49          d //= 2
50          r += 1
51
52      for _ in range(k):
53          a = secrets.randbelow(n - 1) + 1
54          x = pow(a, d, n)
55          if x in (1, n - 1):
56              continue
57          for _ in range(r):
58              x = pow(x, 2, n)
59              if x == 1:
60                  return False
```

```
61              if x == n - 1:
62                  break
63          else:
64              return False
65      return True
66
67
68  def generate_prime(length: int) -> int:
69      '''Generates primes of length bits
70
71      >>> p = generate_prime(32)
72      >>> p.bit_length()
73      32
74
75      p is prime, i.e. not divisible by any numbers up to sqrt(p)
76
77      >>> all(p % n != 0 for n in range(3, 2 ** 16, 2))
78      True
79      '''
80      while True:
81          p = (2 ** (length - 1)) | secrets.randbits(length - 1) |
82          if is_probable_prime(p):
83              return p
84
85
86  def generate_safe_prime(length: int) -> int:
87      '''Generates safe primes of at least `length` bits
88
89      >>> p, q = generate_safe_prime(32)
90      >>> p.bit_length() in [32, 33]
91      True
```

```
92      >>> q.bit_length() in [31, 32]
93      True
94
95      p is safe prime, i.e. in the form 2q+1 where q is a prime number
96
97      >>> all(p % n != 0 for n in range(3, 2 ** 17, 2))
98      True
99      >>> all(q % n != 0 for n in range(3, 2 ** 16, 2))
100     True
101     '''
102     while True:
103         p = generate_prime(length)
104
105         # p1 = 2 * p + 1
106         p1 = (p << 1) | 1
107         if is_probable_prime(p1):
108             return p1, p
109
110         # p1 = (p - 1) // 2
111         p1 = p >> 1
112         if is_probable_prime(p1):
113             return p, p1
114
115
116 def eval_poly(poly, x):
117     '''
118     >>> poly = [4, 3, 2]
119     >>> eval_poly(poly, 0)
120     4
121     >>> eval_poly(poly, 5)
122     69
```

```
123         '''
124         return sum(f * x ** i for i, f in enumerate(poly))
```