Capítulo 2

Sistemas de Votação Fim-a-Fim: Teoria e Prática

Eduardo Lopes Cominetti (USP), Marcos Antonio Simplicio Junior (USP), Paulo Matias (UFSCar), Roberto Samarone Araújo (UFPA)

Abstract

End-to-End voting systems provide greater security guarantees than the ones traditionally used. However, such systems are slightly used in large-scale elections, for positions such as president or governor. This short course aims to present the benefits and challenges of end-to-end voting systems, considering the Brazilian scenario as its potential target. In particular, the discussion is guided by the study of two examples: a system based on mixnets, similar to the first version of the Helios online voting system, implemented with the Verficatum library; and the ElectionGuard system.

Resumo

Sistemas de votação fim-a-fim fornecem maiores garantias de segurança que os tradicionalmente utilizados. No entanto, tais sistemas ainda são pouco utilizados em eleições grande porte, para cargos políticos concorridos como presidente ou governador. Este minicurso tem como objetivo apresentar os benefícios e desafios de sistemas de votação fim-a-fim, considerando o cenário brasileiro como seu potencial alvo de aplicação. Em particular, a discussão é pautada pelo estudo de dois exemplos: um sistema baseado em redes de misturadores, similar à primeira versão do sistema de votação online Helios, implementado com o auxílio da biblioteca Verificatum; e o sistema ElectionGuard.

2.1. Introdução

Atualmente, a democracia é o regime político legalmente mais adotado em escala global. Nesta forma de governo, o povo tem o direito de exercer o poder político, seja de maneira direta ou indireta. A democracia exercida de maneira indireta, ou democracia representativa, requer que o povo seja capaz de eleger seus representantes, que são legitimizados para atuar em seu lugar por um período determinado. Desta maneira, estes representantes,

durante seu mandato, adquirem grandes poderes para criar, promulgar, ou vetar leis, as quais influenciam milhares ou até mesmo milhões de indivíduos. Consequentemente, é de vital importância garantir que o processo de escolha destes representantes, denominado Eleição, seja realizada de maneira integra, refletindo o real desejo da população.

Dentro do processo de Eleição, o sistema de votação é o responsável pela coleta da escolha do eleitor, denominada Voto, bem como por sua guarda, tabulação e, por fim, a apuração dos resultados. Com o intuito de reduzir custos e agilizar o processo de tabulação de votos e apuração, sistemas eletrônicos de votação foram criados. Apesar da ideia de um sistema eletrônico de votação ser conceitualmente simples, ela envolve uma série de desafios, como descritos por Eddie Perez [Perez 2021]. Entre estes desafios, citam-se como exemplos: a quantidade de dados que o sistema trata; a enorme variedade de (falta de) conhecimento técnico que seus usuários possuem; a necessidade de garantir a segurança deste sistema contra ataques tanto de agentes externos como internos; e a necessidade do sistema ser extremamente preciso, mesmo dependendo de equipamentos sujeitos a falhas. Acima de todos estes desafios, é necessário destacar que a passagem pacífica de poder, algo vital em uma democracia, depende da confiança do público no correto funcionamento deste sistema. Nos últimos anos, este último ponto tem sido um desafio particularmente complexo, como observado em movimentos recentes nos Estados Unidos [Berlinski et al. 2021] e no Brasil [Nicas et al. 2022].

Duas das principais desconfianças do público em relação a um sistema eletrônico de votação são referentes à correta coleta do voto do eleitor e sua guarda, de maneira íntegra, e ao correto uso deste voto para a produção do resultado final. Dentre alguns métodos desenvolvidos para assegurar ao eleitor que estes dois processos estão sendo realizados de maneira correta, têm destaque os sistemas de votação fim-a-fim, ou E2E (*end-to-end*). Esses sistemas são capazes de fornecer ao eleitor evidências de que seu voto foi corretamente coletado e gravado pela equipamento eletrônico durante o momento da votação. Posteriormente, esses sistemas também são capazes de fornecer garantias matemáticas de que o voto do eleitor gravado no equipamento eletrônico foi utilizado na apuração da eleição. Portanto, como o voto foi corretamente gravado e, posteriormente, corretamente utilizado na totalização, um sistema E2E é capaz de promover uma maior confiança na integridade do resultado eleitoral.

Um exemplo de solução E2E bastante conhecido, adotado em muitos cenários de votação onde coerção não é considerada uma ameaça, é o sistema de votação online Helios [Adida 2008]. Outras propostas incluem Prêt à Voter [Ryan et al. 2005], Scantegrity II [Chaum et al. 2008], Punchscan [Popoveniuc and Hosp 2010], Remotegrity [Zagórski et al. 2013], VoteXX [Chaum et al. 2021], e ElectionGuard [Benaloh and Naehrig 2022], criando um ecossistema rico e bastante diverso de soluções. Alguns desses sistemas se baseiam em mecanismos criptográficos, comuns especialmente em propostas voltadas a votação online ou presencial, enquanto outros envolvem dispositivos físicos com características específicas, tendo como foco eleições presenciais.

Neste minicurso, são discutidas as principais características e diferenças entre os métodos de votação fim-a-fim baseados em redes de misturadores e criptografia homomórfica, bem como as ferramentas criptográficas utilizadas por cada um deles. Antes da apresentação destes dois sistemas, uma definição e uma breve discussão de um sistema

de votação fim-a-fim são apresentadas na Seção 2.3.1. Em seguida, um breve histórico de sistemas fim-a-fim é apresentado na Seção 2.3.2. A Seção 2.2 apresenta os blocos criptográficos fundamentais que serão utilizados no trabalho. Nas Seções 2.4 e 2.5, os sistemas Helios e ElectionGuard, que são considerados o estado da arte atualmente, são descritos. Na Seção 2.6, é feita uma comparação entre os dois sistemas, destacando-se as vantagens e desvantagens entre eles. Por fim, a Seção 2.7 apresenta a conclusão deste minicurso, juntamente com os desafios futuros em relação a sistemas de votação fim-a-fim.

2.2. Fundamentos de Criptografia

Nesta seção são apresentados os principais componentes criptográficos necessários para a operação dos sistemas fim-a-fim baseados em redes de misturadores e criptografia homomórfica.

2.2.1. Breve definição: chave simétrica e chave assimétrica

Um esquema de criptografia de chave simétrica utiliza uma única chave, ou chaves facilmente correlacionáveis, para tanto cifrar quanto decifrar um determinado texto.

Um esquema de criptografia de chave assimétrica utiliza duas chaves, uma chamada de chave pública, pois é amplamente divulgada, e uma chamada de chave secreta, pois é mantida em sigilo pelo usuário. Além disso, deve ser computacionalmente difícil de se obter a chave secreta a partir da chave pública. A operação no texto feita por uma das chaves apenas pode ser revertida pela outra. Desta maneira, por exemplo, qualquer usuário pode utilizar a chave pública para enviar uma informação que só poderá ser lida pelo dono da respectiva chave secreta. Por outro lado, o dono da chave secreta também pode utilizar a chave secreta para enviar uma informação que pode ter a origem comprovada através do uso da respectiva chave pública.

2.2.2. Breve definição: esquema determinístico e esquema probabilístico

Em um esquema determinístico, a operação de um determinado texto sempre obtém um mesmo resultado. Por exemplo, o simples uso da cifra de bloco AES para cifrar um texto específico sempre irá produzir o mesmo texto cifrado enquanto a chave permanecer inalterada.

Em um esquema probabilístico, a operação de um determinado texto produz resultados diversos, mesmo quando a chave, ou conjunto de chaves, permace inalterada. Isto em geral ocorre pela utilização de um número aleatório único, conhecido como *nonce*, a cada operação do algoritmo. Como exemplo, o esquema de assinatura ECDSA utiliza um ponto aleatório único a cada operação, de modo que ao assinar um mesmo texto duas vezes, duas assinaturas distintas sejam produzidas e ambas possam ser corretamente verificadas.

2.2.3. Criptosistema ElGamal

A cifra de ElGamal [ElGamal 1985] é um esquema de criptografia de chave assimétrica probabilístico proposto por ElGamal em 1985. O esquema tem sua segurança baseada no problema do logaritmo discreto e, portanto, opera através de exponenciações em um grupo cíclico. Mais precisamente, considere o grupo cíclico $\mathbb{Z}_p$, com p primo, onde

$\mathbb{Z}_p = \mathbb{Z} \mod p$. Um gerador g deste grupo cíclico é um número capaz de produzir todos os elementos deste grupo cíclico, com exclusão do elemento nulo, através de sua exponenciação por um número a:

$$\exists x, a \in \mathbb{Z}_p^* : g^a = x$$

A cifra de ElGamal é composta por três operações: geração de chaves, cifração e decifração.

Geração de chaves: Gera o par de chaves secreta **sk** e pública **pk** do sistema. A chave secreta **sk** neste algoritmo é um número aleatório de $\mathbb{Z}_p^*$. Por sua vez, a chave pública **pk** é definida como $\mathbf{pk} = g^{\mathbf{sk}}$.

Cifração ($\mathbf{Enc_{pk}}(m;y)$): Cifra a mensagem $m \in \mathbb{Z}_p$ através do uso da chave pública **pk** e de um número aleatório y, produzindo o texto cifrado $\mathbf{Enc_{pk}}(m;y) = c = (a,b)$, onde:

$$a = m \cdot \mathbf{pk}^y$$
$$b = g^y$$

Decifração ($\mathbf{Dec_{sk}}(c)$): Decifra o texto cifrado $c = (a,b)$ utilizando a chave secreta **sk**, produzindo a mensagem $\mathbf{Dec_{sk}}(c) = m' \in \mathbb{Z}_p$:

$$b' = b^{\mathbf{sk}} = g^{y \cdot \mathbf{sk}} = \mathbf{pk}^y$$
$$m' = a \cdot (b')^{-1} = m \cdot \mathbf{pk}^y \cdot (\mathbf{pk}^y)^{-1} = m$$

2.2.3.1. Propriedade de homomorfismo da cifra de ElGamal

Um leitor atento é capaz de percerber que dados $c_1 = (a_1, b_1)$ e $c_2 = (a_2, b_2)$, que cifram, respectivamente, as mensagens m_1 e m_2, utilizando y_1 e y_2, a multiplicação entre si destes dois textos cifrados produz $c_3 = (a_3, b_3)$, que cifra a mensagem $m_3 = m_1 \cdot m_2$, utilizando $y_3 = y_1 + y_2$. Logo, o esquema de ElGamal possuí um homomorfismo multiplicativo do texto às claras através da multiplicação de textos cifrados:

$$a_3 = a_1 \cdot a_2 = m_1 \cdot \mathbf{pk}^{y_1} \cdot m_2 \cdot \mathbf{pk}^{y_2} = m_1 \cdot m_2 \cdot \mathbf{pk}^{y_1+y_2} = m_3 \cdot \mathbf{pk}^{y_3}$$
$$b_3 = b_1 \cdot b_2 = g^{y_1} \cdot g^{y_2} = g^{y_1+y_2} = g^{y_3}$$

Observa-se também que a decifração também ocorre de maneira correta através do mesmo processo descrito anteriormente:

$$b'_3 = b_3^{\mathbf{sk}} = g^{y_3 \cdot \mathbf{sk}} = \mathbf{pk}^{y_3}$$
$$m'_3 = a_3 \cdot (b'_3)^{-1} = m_3 \cdot \mathbf{pk}^{y_3} \cdot (\mathbf{pk}^{y_3})^{-1} = m_3$$

2.2.3.2. Transformando a propriedade de homomorfismo de multiplicação para soma: ElGamal Exponencial

Apesar do homomorfismo multiplicativo ser útil em algumas situações, em um cenário de votação, o homomorfismo de soma, ou seja, a operação nos textos cifrados resulta em

uma soma dos textos às claras, é o ideal. Isto se deve ao fato de se desejar utilizar o esquema para somar os votos para produzir o resultado da eleição.

Felizmente, é possível transformar o homomorfismo multiplicativo da cifra de ElGamal para um homomorfismo de soma, contanto que as mensagens a serem cifradas sejam pequenas. Para tanto, ao invés de simplesmente cifrar as mensagens m_i, um usuário cifra as mensagens $m'_i = g^{m_i}$. Consequentemente, a multiplicação de c_1 por c_2 produz c_3, que cifra a mensagem $m'_3 = g^{m_1} \cdot g^{m_2} = g^{m_1+m_2} = g^{m_3}$. Esta variante da cifra de ElGamal recebe o nome de ElGamal Exponencial.

É importante observar que, após a decifração de c_3, é necessário calcular o logaritmo discreto da mensagem m'_3 resultante para a obtenção da soma das mensagens m_1 e m_2. Apesar do cálculo do logaritmo discreto ser um problema computacionalmente difícil, ele é possível de ser realizado contanto que as mensagens tenham um tamanho relativamente pequeno. Por exemplo, a eleição brasileira conta com cerca de 150 milhões de eleitores cadastrados, o que corresponde a uma mensagem de tamanho máximo inferior a 2^{28}. Este tamanho de mensagem ainda é considerado pequeno e apto para ser computacionalmente calculável ou mantido em uma tabela para consulta. Logo, a aplicação do ElGamal Exponencial é viável para o cenário eleitoral.

2.2.3.3. Recifração de texto cifrado por ElGamal

É possível utilizar a propriedade de homomorfismo em ambas as variantes da cifra de ElGamal para recifrar um determinado texto cifrado c_1, produzindo $c_3 \neq c_1$, mas com $m_3 = m_1$. Este fato é útil, pois é inviável para um observador externo relacionar c_1 e c_3. Em outras palavras, caso um observador veja três mensagens, c_1, c'_1 e c_3, ele não é capaz de dizer se c_3 foi produzido utilizando c_1, c'_1,ou até mesmo um terceiro c''_1 desconhecido. Este fato será utilizado nas provas de conhecimento zero relacionadas a redes de misturadores apresentadas futuramente neste texto.

Para realizar a recifração, basta escolher y_2 aleatório e calcular a cifração da mensagem $m_2 = 1$, ou $m'_2 = 1$ para o ElGamal Exponencial, obtendo c_2. Posteriormente, basta realizar a multiplicação de c_1 e c_2, obtendo-se c_3:

$$a_3 = a_1 \cdot a_2 = m_1 \cdot \mathbf{pk}^{y_1} \cdot 1 \cdot \mathbf{pk}^{y_2} = m_1 \cdot \mathbf{pk}^{y_1+y_2} = m_3 \cdot \mathbf{pk}^{y_3}$$
$$b_3 = b_1 \cdot b_2 = g^{y_1} \cdot g^{y_2} = g^{y_1+y_2} = g^{y_3}$$

2.2.3.4. ElGamal com decifração limiar (*Threshold ElGamal*)

Ao invés do par de chaves na cifra de ElGamal ser constituída por uma única chave secreta, é possível criar um conjunto de chaves secretas e uma única chave pública associada. Este é um fato interessante, pois enquanto a cifração de uma mensagem é executada de maneira análoga a construção com chave única, a decifração do texto cifrado requer um número mínimo pré-estabelecido destas chaves para ser realizada. A esta variante que possuí múltiplas chaves secretas com a necessidade de uma quantidade mínima delas para operar a decifração, é dado o nome de ElGamal com decifração limiar (*Threshold ElGamal*).

Primeiramente, é estabelecido quantas chaves existirão no sistema ao todo, n, e quantas são necessárias para realizar a decifração, k. Em geral, cada chave é criada por uma entidade independente. Para cada entidade i, temos que a chave secreta $\mathbf{sk}_i$ é um número aleatório de $\mathbb{Z}_p^*$, e a chave pública é definida como $\mathbf{pk}_i = g^{\mathbf{sk}_i}$. Por sua vez, a chave pública combinada é simplesmente o produtório de todas as chaves públicas individuais:

$$\mathbf{pk} = \prod_{i=1}^{n} \mathbf{pk}_i$$

Como segundo passo, a entidade i cria um polinômio P_i de grau $k-1$, sendo o coeficiente de grau 0 igual a chave secreta $\mathbf{sk}_i$ e os demais coeficientes $a_{i,j}$ escolhidos aleatoriamente em $\mathbb{Z}_p^*$:

$$P_i(x) = \prod_{j=1}^{k-1} a_{i,j} \cdot x^j$$

Para cada outra entidade ℓ, a entidade i calcula $P_i(\ell)$ e envia-o para ℓ.

Por fim, para decifrar um texto cifrado usando este esquema de chaves, temos dois casos: quando todas as entidades estão presentes e quando apenas o limiar delas encontra-se presente.

Caso todas elas estejam presentes, e lembrando que $\mathbf{Enc}_{\mathbf{pk}}(m;r) = (a,b) = (m \cdot \mathbf{pk}^r, g^r)$, cada entidade calcula $b^{\mathbf{sk}_i}$ e publica este resultado. Basta então combinar todos estes fatores através de um produtório, inverter o resultado e multiplicá-lo por a, obtendo a mensagem m:

$$a \cdot (\prod_{i=1}^{n} b^{\mathbf{sk}_i})^{-1} = m \cdot \mathbf{pk}^r \cdot (\prod_{i=1}^{n} g^{r \cdot \mathbf{sk}_i})^{-1} = m \cdot \mathbf{pk}^r \cdot ((g^{\sum_{i=1}^{n} \mathbf{sk}_i})^r)^{-1} =$$
$$= m \cdot \mathbf{pk}^r \cdot (\mathbf{pk}^r)^{-1} = m$$

Caso apenas um limiar destas entidades estejam presentes, estas entidades utilizam o resultado do polinomio enviado anteriormente pelas entidades faltantes para decifrar o texto cifrado. Isto é feito através do cálculo de parcelas do valor que deveria ser retornado pelas entidades ausentes. Caso a entidade i não esteja presente, uma entidade ℓ prossegue calculando e publicando o valor parcial $M_{i,\ell}$:

$$M_{i,\ell} = b^{P_i(\ell)}$$

Então, é calculado o valor do coeficiente de Lagrange para um conjunto de k entidades disponíveis $\{\ell \in U, |U| = k\}$:

$$\omega_\ell = \prod_{j \in (U - \{\ell\})} \frac{j}{j - \ell}$$

Por fim, o valor final referente a entidade ausente é computado como:

$$M_i = \prod_{\ell \in U} (M_{i,\ell})^{\omega_\ell}$$

Este valor M_i é utilizado no lugar de $b^{\mathbf{sk}_i}$ e a decifração é realizada como se todas as entidades estivessem presentes.

2.2.4. Esquema de Compromisso de Pedersen

O compromisso de Pedersen é um esquema baseado em um grupo cíclico. Um esquema de compromisso é um algoritmo que permite que um usuário se comprometa com uma ou mais mensagens perante outros usuários, mas sem revelar os seus valores. Posteriormente, o valor das mensagens pode ser revelado e pode-se averiguar que eles são condizentes com o compromisso previamente informado. O compromisso de Pedersen possuí a propriedade de esconder de forma perfeita ("*perfectly hiding*") e a propriedade de ligação de forma computacional ("*computationally binding*").

No compromisso de Pedersen, deseja-se criar um compromisso para n mensagens. Para tanto, é necessário escolher $n+1$ geradores indepentes do grupo cíclico, $g, h_1, h_2, \ldots, h_n$. Geradores independentes são geradores cuja relação entre eles é desconhecida, ou seja, não se conhece qual o valor do logaritmo discreto de um gerador em relação a outro. Estes geradores são tornados públicos.

Após a escolha dos geradores, um número aletório único r é escolhido o compromisso de Pedersen das n mensagens $\mathbf{m} = (m_1, m_2, \ldots, m_n)$, $\mathbf{Com}(\mathbf{m}; r)$, pode ser calculado por:

$$\mathbf{Com}(\mathbf{m}; r) = g^r \prod_{i=1}^{n} h_i^{m_i}$$

Observa-se que caso o grupo de mensagens $\mathbf{m}$ possua apenas uma mensagem x não nula, cujo valor seja $m_x = 1$, o compromisso pode ser simplesmente calculado por:

$$\mathbf{Com}(\mathbf{m}; r) = g^r \cdot h_x$$

Este fato será utilizado nas provas de conhecimento zero relacionadas a redes de misturadores apresentadas futuramente neste texto.

2.2.5. Provas de Conhecimento Não Interativas

Prova de conhecimento são em geral interativas, ou seja, elas requerem que o indivíduo que está realizando a prova interaja com o verificador durante a sua execução. Por meio da heuristica de Fiat-Shamir [Fiat and Shamir 1986], no entanto, provas de conhecimento zero interativas podem ser convertidas em provas não interativas. Dessa forma, o realizador da prova não precisa estar disponível no momento da verificação da prova. A seguir são apresentadas três provas de conhecimento não interativas utilizadas nesse minicurso.

2.2.5.1. Prova de Chaum-Pedersen

A prova de conhecimento zero de Chaum-Pedersen [Chaum and Pedersen 1992] permite provar o valor do texto às claras de um texto cifrado com ElGamal. Para isto, é necessário o conhecimento do número aleatório utilizado na cifração ou o conhecimento da chave privada.

Considere o criptosistema de ElGamal Exponencial, definido no grupo cíclico $\mathbb{Z}_p$,com p primo. Adicionalmente, considere o subgrupo cíclico de resíduos r deste grupo, $\mathbb{Z}_p^r = \{y \in \mathbb{Z}_p^*, \exists x \in \mathbb{Z}_p^* : y = x^r \mod p\}$, e seu gerador g. Por fim, seja o primo $q =$

$(p-1)/r$ e o máximo divisor comum entre q e r seja 1. Nesta configuração, a mensagem m a ser cifrada pertence ao subgrupo cíclico de resíduos r deste grupo, $\mathbb{Z}_p^r$ e o número aleatório a ser utilizado pertence a $\mathbb{Z}_q$.

Para a prova utilizando o conhecimento do número aleatório r, considerando o texto cifrado $\mathbf{Enc_{pk}}(m,r) = (g^m \cdot \mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe um valor aleatório $u \in \mathbb{Z}_q$ e cria um compromisso deste valor através da tupla $(a',b') = (\mathbf{pk}^u, g^u)$. Em seguida, cria-se um desafio c, utilizando-se o método de Fiat-Shamir, com $c = H(a,b,a',b')$, sendo H uma função de hash criptográfico. Por fim, o provador calcula $v = u + c \cdot r$ e publica os valores (a',b',v)

O verificador, por sua vez, consegue confirmar a prova através da checagem das seguintes igualdades:

$$g^{m \cdot c} \cdot \mathbf{pk}^v \equiv a' \cdot a^c$$
$$g^v \equiv b' \cdot b^c$$

Para a prova utilizando o conhecimento da chave secreta **sk**, considerando o texto cifrado $\mathbf{Enc_{pk}}(m,r) = (g^m \cdot \mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe um valor aleatório $u \in \mathbb{Z}_q$ e cria um compromisso deste valor através da tupla $(a',b') = (b^u, g^u)$. Em seguida, o provador calcula $d = b^{\mathbf{sk}}$ e cria um desafio c, com $c = H(a,b,a',b',d)$, de forma análoga a anterior. Por fim, o provador calcula $v = u + c \cdot \mathbf{sk}$ e publica os valores (a',b',d,v)

O verificador, por sua vez, consegue confirmar a prova através da checagem das seguintes igualdades:

$$g^v \equiv b' \cdot \mathbf{pk}^c$$
$$b^v \equiv a' \cdot d^c$$

2.2.5.2. Prova de Cramer-Damgård-Schoenmakers

A técnica de Cramer-Damgård-Schoenmakers [Cramer et al. 1994] permite provar a disjunção de dois predicados. Isto significa que dado duas provas de conhecimento zero referentes a um objeto, contraditórias entre si, uma delas é verdadeira, mas não se sabe qual.

Para isto, a técnica utiliza o fato de ser possível construir provas que verifiquem corretamente para afirmações falsas caso o desafio a ser escolhido seja previamente conhecido. Para garantir que em duas provas uma delas possua um valor pré-definido pelo provador e outra possua um valor aleatório, utiliza-se uma composição de um desafio global aleatório. Caso um desafio global c seja escolhido utilizando-se o método de Fiat-Shamir, é possível escolher um valor c_0 para um dos desafios, enquanto o outro desafio c_1 corresponde ao complemento deste valor em relação a c, ou seja, $c = c_0 + c_1$. Consequentemente, apesar de c_0 ser definido pelo provador, o seu complemento c_1 em relação a um aleatório c continua aleatório.

Neste minicurso, a técnica de Cramer-Damgård-Schoenmakers é utilizada para provar que um texto cifrado com ElGamal Exponencial corresponde a mensagem 0 ou a mensagem 1, tendo conhecimento do valor aleatório r. As construções para os dois casos, um com uma mensagem verdadeiramente 0 e um com uma mensagem verdadeiramente 1 são mostradas a seguir.

Para uma **mensagem de valor 0**, representada por $\mathbf{Enc_{pk}}(0,r) = (\mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe $u,v,w \in \mathbb{Z}_q$ e cria dois compromissos:

$$(a_0,b_0) = (\mathbf{pk}^u, g^u)$$
$$(a_1,b_1) = (g^w \cdot \mathbf{pk}^v, g^v)$$

Em seguida, o valor do desafio global é definido através do cálculo de $c = H(a,b,a_0,b_0, a_1,b_1)$, seguindo a heurística de Fiat-Shamir. Por fim, o provador calcula os seguintes valores e publica (c_0,c_1,v_0,v_1):

$$c_0 = c - w$$
$$c_1 = w$$
$$v_0 = u + c_0 \cdot r$$
$$v_1 = v + c_1 \cdot r$$

O verificador, por sua vez, é capaz de confirmar o conjunto de provas validando as igualdades:

$$c = c_0 + c_1$$
$$(a_0 \cdot a^{c_0}, b_0 \cdot b^{c_0}) \equiv (\mathbf{pk}^u \cdot \mathbf{pk}^{c_0 \cdot r}, g^u \cdot g^{c_0 \cdot r}) \equiv (\mathbf{pk}^{v_0}, g^{v_0})$$
$$(a_1 \cdot a^{c_1}, b_1 \cdot b^{c_1}) \equiv (g^w \cdot \mathbf{pk}^v \cdot \mathbf{pk}^{c_1 \cdot r}, g^v \cdot g^{c_1 \cdot r}) \equiv (g^{c_1} \cdot \mathbf{pk}^{v_1}, g^{v_1})$$

Para uma **mensagem de valor 1**, representada por $\mathbf{Enc_{pk}}(1,r) = (g \cdot \mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe $u,v,w \in \mathbb{Z}_q$ e cria dois compromissos:

$$(a_0,b_0) = (g^w \cdot \mathbf{pk}^v, g^v)$$
$$(a_1,b_1) = (\mathbf{pk}^u, g^u)$$

Em seguida, o valor do desafio global é definido através do cálculo de $c = H(a,b,a_0,b_0, a_1,b_1)$, seguindo a heurística de Fiat-Shamir. Por fim, o provador calcula os seguintes valores e publica (c_0,c_1,v_0,v_1):

$$c_0 = q - w$$
$$c_1 = c + w$$
$$v_0 = v + c_0 \cdot r$$
$$v_1 = u + c_1 \cdot r$$

O verificador, por sua vez, é capaz de confirmar o conjunto de provas validando as igualdades:

$$c = c_0 + c_1$$
$$(a_0 \cdot a^{c_0}, b_0 \cdot b^{c_0}) \equiv (g^w \cdot \mathbf{pk}^v \cdot g^{c_0} \cdot \mathbf{pk}^{c_0 \cdot r}, g^v \cdot g^{c_0 \cdot r}) \equiv (\mathbf{pk}^{v_0}, g^{v_0})$$
$$(a_1 \cdot a^{c_1}, b_1 \cdot b^{c_1}) \equiv (\mathbf{pk}^u \cdot g^{c_1} \cdot \mathbf{pk}^{c_1 \cdot r}, g^u \cdot g^{c_1 \cdot r}) \equiv (g^{c_1} \cdot \mathbf{pk}^{v_1}, g^{v_1})$$

2.2.5.3. Prova de embaralhamento de Wikström-Terelius

A prova de conhecimento zero de embaralhamento de Wikström-Terelius [Terelius and Wikström 2010] busca provar que duas listas de textos cifrados distintas, $\mathbf{e} = (e_1, e_2, \ldots, e_n)$

e $\mathbf{e}' = (e'_1, e'_2, \ldots, e'_n)$, contém os mesmos textos às claras $\mathbf{m} = (m_1, m_2, \ldots, m_n)$, mas em ordem permutada.

Para isto, considere a permutação $\psi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ e a matriz de permutação de $\mathbf{B}_\psi = (b_{ij})_{n \times n}$, onde:

$$b_{ij} = \begin{cases} 1, & \text{se } \psi(i) = j, \\ 0, & \text{caso contrário.} \end{cases}$$

Ou seja, a coluna j da linha i é igual a 1 caso e'_j represente e_i e 0 caso contrário. A lista $\mathbf{e}$ é então multiplicada por esta matriz e os elementos resultantes são recifrados, produzindo a lista $\mathbf{e}'$. Desta maneira, é necessário provar que: 1) a matriz de permutação possui apenas um único elemento 1 em cada linha e em cada coluna; 2) a lista original $\mathbf{e}$ foi multiplicada por esta matriz; e 3) a lista $\mathbf{e}'$ é a recifração dos elementos do resultado de 2. Para construir estas provas será utilizado o compromisso de Pedersen previamente apresentado.

Sendo a coluna da matriz representada por $\mathbf{b_j} = (b_{1,j}, \ldots, b_{n,j})$, temos que o compromisso de cada coluna com aleatoriedade r_j é dado por:

$$\mathbf{Com}(\mathbf{b_j}; r_j) = g^{r_j} \prod_{i=1}^{n} h_i^{b_{ij}} = g^{r_j} \cdot h_i$$

Adicionalmente, considerando que $\mathbf{r} = (r_1, \ldots, r_n)$, adota-se a seguinte notação como o compromisso da matriz de permutação:

$$\mathbf{Com}(\psi; \mathbf{r}) = (\mathbf{Com}(\mathbf{b_1}; r_1), \ldots, \mathbf{Com}(\mathbf{b_n}; r_n)) = (c_1, \ldots, c_n) = \mathbf{c}$$

Para provar que a matriz de permutação $\mathbf{B}_\psi = (b_{ij})_{n \times n}$ possui apenas um único elemento 1 em cada linha e cada coluna, considere uma lista auxiliar de elementos aleatórios $\mathbf{x} = (x_1, \ldots, x_n)$. A afirmação é verdadeira se e somente se:

$$\sum_{j=1}^{n} b_{ij} = 1, \text{para } 1 \leq i \leq n$$
$$\prod_{i=1}^{n} \sum_{j=1}^{n} b_{ij} \cdot x_i = \prod_{i=1}^{n} x_i$$

Lembrando que $\mathbf{c}$ corresponde ao compromisso da matriz de permutação e considerando $\mathbf{r} = (r_1, \ldots, r_n)$ e $\bar{r} = \sum_{j=1}^{n} r_j$, é possível observar que podemos representar a primeira condição como (notar que a somatória de b_{ij} aparece como exponente de h_i):

$$\prod_{j=1}^{n} c_j = \prod_{i=1}^{n} g^{r_j} \prod_{i=1}^{n} h_i^{b_{ij}} = g^{\sum_{j=1}^{n} r_j} \prod_{i=1}^{n} h_i^{\sum_{j=1}^{n} b_{ij}} = g^{\bar{r}} \prod_{i=1}^{n} h_i = \mathbf{Com}(\mathbf{1}, \bar{r}) \tag{1}$$

Por sua vez, considerando uma lista de valores arbitrários $\mathbf{u} = (u_1, \ldots, u_n)$ e o resultado de sua multiplicação pela matriz de permutação $\mathbf{u}' = (u'_1, \ldots, u'_n)$, e sendo $\tilde{r} = \sum_{j=1}^{n} r_j \cdot u_j$, a segunda condição pode ser representada pelas equações (notar que o uso de c_j relaciona as duas próximas equações com a anterior):

$$\prod_{i=1}^{n} u_i = \prod_{i=1}^{n} u'_i \tag{2}$$

$$\prod_{j=1}^{n} c_j^{u_j} = \prod_{j=1}^{n} (g^{r_j} \prod_{i=1}^{n} h_i^{b_{ij}})^{u_j} = g^{\sum_{j=1}^{n} r_j \cdot u_j} \prod_{i=1}^{n} h_i^{\sum_{j=1}^{n} b_{ij} \cdot u_j} = g^{\tilde{r}} \prod_{i=1}^{n} h_i^{u'_i} = \\ = \mathbf{Com}(\mathbf{u}', \tilde{r}) \tag{3}$$

Por fim, recorda-se que $\mathbf{e}'$ é a lista permutada e recifrada da lista original $\mathbf{e}$, que estas listas tem como elementos textos cifrados utilizando ElGamal e que uma recifração em ElGamal é obtida através da multiplicação do texto cifrado por uma cifração do valor 1 utilizando um novo valor aleatório. Sendo o valor aleatório r'_j o valor empregado em cada e_j para produzir seu elemento equivalente na lista $\mathbf{e}'$, utilizando os valores $\mathbf{u}$ e $\mathbf{u}'$ definidos anteriormente, e sendo $r' = \sum_{j=1}^{n} = r'_j \cdot u_j$, para provar que $\mathbf{e}'$ é uma recifração de $\mathbf{e}$, temos:

$$\prod_{i=1}^{n} (e'_i)^{u'_i} = \prod_{j=1}^{n} (e_j \cdot \mathbf{Enc}_{pk}(1; r'_j))^{u_j} = \prod_{j=1}^{n} (e_j^{u_j} \cdot \mathbf{Enc}_{pk}(1; r'_j)^{u_j}) = \\ = \prod_{j=1}^{n} (e_j^{u_j} \cdot \mathbf{Enc}_{pk}(1; r'_j \cdot u_j)) = \mathbf{Enc}_{pk}(1; r') \cdot \prod_{j=1}^{n} e_j^{u_j} \tag{4}$$

Assim, a prova de conhecimento zero de embaralhamento de Wikström-Terelius consiste em demonstrar, utilizando uma prova de conhecimento não interativa de pré-imagem, estas quatro equações:

$$\text{NIZKP} \left[(\bar{r}, \tilde{r}, r', \mathbf{u}') : \begin{array}{l} \prod_{j=1}^{n} c_j = \mathbf{Com}(\mathbf{1}, \bar{r}) \\ \wedge \prod_{i=1}^{n} u_i = \prod_{i=1}^{n} u'_i \\ \wedge \prod_{j=1}^{n} c_j^{u_j} = \mathbf{Com}(\mathbf{u}', \tilde{r}) \\ \wedge \prod_{i=1}^{n} (e'_i)^{u'_i} = \mathbf{Enc}_{pk}(1; r') \cdot \prod_{j=1}^{n} e_j^{u_j} \end{array} \right]$$

Por questões de espaço, a transformação destas equações em provas de conhecimento zero não é mostrada neste minicurso. Entretanto, esta transformação, junto com um pseudo-código para sua implementação, encontra-se descrita no artigo [Haenni et al. 2017].

2.3. Sistemas de Votação Fim-a-Fim (E2E) e sua Evolução Histórica

Esta Seção apresenta as características de um sistema de votação fim-a-fim e como estas características buscam atender as duas principais propriedades desejadas de um sistema de votação: o **sigilo** do voto e a **integridade** do voto e do resultado.

Também são discutidas as diferenças entre o modelo de votação fim-a-fim e o mecanismo de Trilha em Papel para Auditoria Verificada pelo Eleitor (do inglês *Voter Verified Paper Audit Trail*, ou VVPAT).

Por fim, são apresentados alguns dos sistemas de votação que implementam a abordagem fim-a-fim, tanto de maneira mecânica (votação em papel) quanto de maneira eletrônica.

2.3.1. Sistemas de Votação Fim-a-Fim (E2E)

De acordo com Comissão de Assistência Eleitoral Amedicana (EAC) [Commission 2023], um sistema de votação fim-a-fim (E2E) é um sistema que permite com que eleitores individuais sejam capazes de verificar elementos cruciais de uma apuração eleitoral sem que sejam necessária a confiança no software de votação, no hardware de votação, em servidores do órgão eleitoral, ou até mesmo em observadores externos. Para atingir este objetivo, um sistema de votação E2E deve fornecer as seguintes propriedades:

- Voto é lançado como pretendido pelo eleitor – permite que o eleitor verifique que seu voto foi corretamente interpretado pelo sistema de votação enquanto estiver no local de votação;
- Voto é gravado como foi lançado - permite que o eleitor verifique que seu voto foi corretamente gravado pelo sistema de votação e incluído em um registro de votos público;
- Voto é apurado como foi gravado - o sistema eleitoral providencia um método público de apuração que utiliza o registro de votos público.

Com estas propriedades, cada eleitor é capaz de verificar, de maneira individual, que sua escolha de candidato foi corretamente salva pelo sistema e posteriormente corretamente utilizada pelo sistema para produzir a apuração da eleição. Portanto, um sistema E2E é capaz de prover garantias referentes à **integridade** do processo eleitoral.

Para satisfazer estas três propriedades E2E, os dois sistemas de enfoque deste trabalho adotam procedimentos similares, embora com o uso de técnicas diferentes. Consequentemente, nesta Seção a descrição destes procedimentos será feita de maneira única e as técnicas utilizadas para a suas implementações serão aprofundadas nas respectivas seções individuais para cada método.

2.3.1.1. Voto é lançado como pretendido pelo eleitor

Primeiramente, para prover garantias que o voto é lançado como pretendido pelo eleitor, o sistema E2E cifra este voto, utilizando um esquema assimétrico probabilístico de cifração. Na sequência, é computado um hash criptográfico deste valor cifrado, juntamente com alguns dados adicionais públicos. O resultado desta operação, chamado de **Código de Rastreio**, por razões que serão explicadas adiante, é então fornecido ao eleitor em conjunto com os dados adicionais (e.g., através da impressão de um código QR). Até o presente momento, o sistema ainda não realizou a gravação do voto do eleitor. Antes de realizar a gravação do voto, o sistema fornece ao eleitor a capacidade de "desafiar" o voto inserido, procedimento conhecido na literatura como "Desafio de Benaloh" [Benaloh 2006]. O eleitor então possui duas escolhas: 1) desafiar o voto associado ao Código

de Rastreio previamente fornecido; ou 2) permitir que o sistema salve (deposite) o voto cifrado associado ao Código de Rastreio previamente fornecido.

Caso o eleitor opte pelo desafio, o sistema fim-a-fim fornece ao eleitor uma informação extra (i.e., o número único utilizado para realizar a cifração do voto). Esta informação permite que o eleitor, em um dispositivo de sua posse, executando um programa de verificação fornecido por alguma entidade na qual ele confie (ou até mesmo um programa escrito por ele próprio), realize a verificação de correta construção do Código de Rastreio. Para isto, basta que o programa repita os passos do sistema de votação: com o voto do eleitor e o número aleatório utilizado originalmente na cifração, o verificador cifra o voto utilizando o esquema assimétrico e computa seu hash criptográfico. Realizado este procedimento, o programa então compara o resultado do hash que ele computou com o Código de Rastreio fornecido pelo sistema. Caso a comparação seja bem sucedida, o sistema originalmente comportou-se de maneira honesta e correta e utilizou o voto fornecido pelo eleitor. Esta afirmação é possível de ser feita pois para o sistema fornecer uma informação adicional que gere o mesmo Código de Rastreio para um voto diferente do inserido pelo eleitor, o sistema teria que ser capaz de resolver o problema matemático de segurança criptográfica associado ao algoritmo de cifração ou ser capaz de calcular a segunda pré-imagem do hash criptográfico utilizado.

Adicionalmente, uma vez que o eleitor opte pelo desafio, o sistema E2E apaga o voto do eleitor e todos os dados gerados e exige que o eleitor reinicie o processo de votação. Desta forma, toda vez que um desafio é realizado, o eleitor precisa reinserir seu voto no sistema e um novo Código de Rastreio, totalmente independente do Código de Rastreio anterior, é gerado. Isto é realizado para garantir o sigilo do voto do eleitor, afinal, caso o eleitor pudesse gravar um voto já desafiado, ele seria capaz de provar para qualquer pessoa em qual candidato ele votou. Consequentemente, o Desafio de Banaloh é um processo que fornece uma garantia estatística ao eleitor de que seu voto foi lançado como pretendido. Como o sistema não sabe se poderá gravar ou não o voto do eleitor antes de lhe fornecer o Código de Rastreio, caso o sistema se comporte de maneira errônea, haverá uma possibilidade do erro ser detectado. Uma análise simplificada desta garantia estatística é apresentada na Seção 2.3.1.5.

2.3.1.2. Voto é gravado como foi lançado

Para fornecer uma prova de que o voto é gravado como foi lançado, uma vez que o eleitor permita com que o sistema salve o voto associado ao Código de Rastreio, o sistema assina digitalmente o Código de Rastreio fornecido ao eleitor.

Uma assinatura digital é um algoritmo criptográfico que fornece o serviço de irretratabilidade a um sistema. Desta maneira, uma vez que o Código de Rastreio seja assinado pelo sistema fim-a-fim, o eleitor é capaz de provar, além de uma dúvida razoável, de que o voto cifrado associado a este Código de Rastreio deveria constar no registro de votos do sistema. Adicionalmente, está é a razão pelo nome do Código de Rastreio: através dele é possível rastrear que o voto cifrado do eleitor encontra-se no registro público da eleição. Como ele permite rastrear apenas o voto cifrado, o Código de Rastreio não compromete o sigilo do voto do eleitor.

Por fim, é importante notar que um eleitor malicioso não é capaz de produzir um Código de Rastreio falso para alegar que um possível voto não consta no registro público do sistema. Embora todas as etapas de criação do Código de Rastreio possam ser reproduzidas com sucesso fora do sistema, o eleitor malicioso não é capaz de gerar uma assinatura digital válida para este Código de Rastreio. Consequentemente, este tipo de ataque é mitigado em um sistema fim-a-fim.

2.3.1.3. Voto é apurado como foi gravado

Por fim, para fornecer uma prova de que o voto é apurado como foi gravado, um sistema E2E demonstra matematicamente que os votos cifrados no registro público foram utilizados na soma do resultado. Este processo envolve a decifração dos votos individuais ou de um agregado de votos cifrados somados homomorficamente, sem que esta decifração comprometa o sigilo do voto. A maneira que esta etapa é realizada é o principal diferencial entre os dois métodos estudados neste minicurso.

Como esperado, um sistema baseado em rede de misturadores embaralha os votos cifrados antes de decifrá-los, provando que o embaralhamento foi feito de maneira correta (i.e., nenhum voto foi removido ou inserido). Uma vez que os votos são decifrados, a apuração do resultado ocorre através do simples tabulamento e soma dos votos.

Por sua vez, um sistema baseado em criptografia homomórfica simplesmente soma todos os votos cifrados e realiza a decifração do agregado de votos. Desta maneira, uma vez que a decifração ocorra, o resultado final da eleição já encontra-se calculado.

Em ambos os casos, o sistema também prova que a decifração dos votos foi feita de maneira correta.

2.3.1.4. Sobre o Código de Rastreio

Como mencionado anteriormente, o Código de Rastreio é o resultado de um hash criptográfico que inclui o voto cifrado e alguns dados adicionais. Estes dados adicionais são o Código de Rastreio do último voto gravado no sistema, criando uma cadeia de hashes dos Códigos de Rastreio, e o instante de tempo no qual o eleitor inseriu o voto no sistema. Estes dados são incluídos no calculo do Código de Rastreio para prevenir dois ataques: o ataque de preplay do voto cifrado e um ataque de descarte do Código de Rastreio.

Um ataque de preplay permite com que o sistema desvie votos. Para isto, o sistema cria um voto cifrado para o candidato A, cria o Código de Rastreio para este voto cifrado, assina-o e lança este voto no sistema. Este Código de Rastreio, então, seria exibido para múltiplos eleitores. Sempre que um eleitor fizesse a escolha por candidato A, o sistema apresentaria esse Código de Rastreio para o eleitor. Caso o eleitor deposite este voto, a urna entrega a assinatura do Código de Rastreio para o eleitor, mas ela é capaz de criar um novo voto, com qualquer escolha de candidato, e depositá-lo no lugar do voto do eleitor. Caso o eleitor verifique o registro de votos público, ele será capaz de encontrar o Código de Rastreio provido pelo sistema e ver que o voto associado a este Código de Rastreio foi utilizado na apuração. Desta maneira, o sistema é capaz de criar um único voto que será

utilizado por todos os eleitores que fizerem a escolha do candidato A, enquanto desvia votos para outros candidatos.

Em um ataque de descarte do Código de Rastreio, votos no sistema podem ser substituídos. No momento anterior ao encerramento da votação, uma pessoa poderia procurar por todos os Códigos de Rastreio descartados pelos eleitores, como por exemplo, os Códigos de Rastreio que eleitores desinteressados em verificar o resultado jogaram no lixo. Pelo fato destes Códigos de Rastreio estarem descartados, sua validade nunca será checada perante o registro de votos público. Desta forma, estes Códigos de Rastreio podem ser apagados e substituídos no sistema. Uma vez realizada a substituição, a votação é encerrada normalmente, produzindo o registro público e as provas matemáticas de que os votos cifrados, agora alterados, foram utilizados para produzir o resultado.

O uso do Código de Rastreio em uma cadeia de hashes, que inclui o Código de Rastreio anterior, o instante de inserção do voto no sistema e o voto cifrado, previne ambos os ataques. Como o Código de Rastreio é criado usando o instante de inserção do voto no sistema, é possível verificar, ao receber o Código de Rastreio, que ele foi recém criado. Consequentemente, o sistema não conseguiria apresentar para múltiplos eleitores o mesmo Código de Rastreio, pois o eleitor conseguiria verificar que o instante de tempo presente no Código de Rastreio difere significativamente do momento atual.

Analogamente, o Código de Rastreio em uma cadeia de hashes previne o ataque de descarte do Código de Rastreio. Uma vez que o Código de Rastreio se encontra em uma cadeia de hashes, uma substituição de um Código de Rastreio descartado necessitaria que o sistema fosse capaz de substituir um elemento da cadeia sem com que os próximos elementos fossem afetados, o que é implausível. Esta é a mesma característica que garante que transações feitas em Blockchains não possam ser modificadas.

2.3.1.5. Análise do desafio de Benaloh

A seguir são apresentadas as análises estatísticas referente ao processo de desafio de Benaloh e a simulação de seu uso em casos reais.

Como mencionado anteriormente, o desafio de Benaloh é o processo estatístico que permite ao eleitor verificar que seu voto foi lançado como pretendido, ou seja, que o voto inserido no sistema representa o desejo do eleitor. O processo é considerado estatístico, pois o sistema não sabe se será ou não desafiado e, portanto, um eventual erro pode não ser detectado.

Entretanto, como será apresentado na análise estatística a seguir, o número de desafios que devem ocorrer no sistema para que eventuais erros não impactem no resultado da eleição é pequeno. Essa análise é aplicada, adicionalmente, a alguns cenários reais para confirmar a utilidade do mecanismo.

Para a análise estatística, nós consideramos uma eleição simples entre dois candidatos, A e B, onde o candidato A é o vencedor. Neste cenário, existem V_T eleitores e A recebe a dos votos, com $0,5 < a \leq 1$. Consequentemente, o candidato B recebe $(1-a) \cdot V_T$ dos votos. Adicionalmente, s dos votos são desafiados pelos eleitores, sendo o desafio incondicional a escolha do eleitor (i.e., os eleitores de A e B desafiam o sistema

com a mesma probabilidade).

Para o resultado desta eleição ser alterado, é necessário que o sistema desvie votos do candidato A para o candidato B. A análise considera que o sistema é poderoso o suficiente para desviar a quantidade mínima de votos para que o resultado da eleição mude. Em outras palavras, o sistema muda votos de A para B de tal modo que, ao final do processo, B ganhe a eleição por um único voto.

Para que isto ocorra, a urna deve desviar d votos de A para B, onde:

$$d = \frac{V_T \cdot (2 \cdot a - 1) + 1}{2 \cdot a \cdot V_T}$$

Uma vez que apenas votos de A são desviados, a probabilidade l de um voto ser desviado é $l = a \cdot d$, e a probabilidade deste desvio ser detectado c é a probabilidade de um voto desviado ser desafiado, ou seja, $c = l \cdot s$.

Finalmente, como um voto desafiado é destruído e o eleitor deve realizar novamente o fluxo de votação, onde o novo voto pode ser novamente desafiado com a mesma probabilidade s, o sistema deve criar, no total, $V_C = V_T/(1-s)$ votos.

Para que um sistema desonesto tenha sucesso em sua tentativa de alterar o resultado eleitoral, nenhum dos votos desviados pode ser detectado. A probabilidade deste fato ocorrer é dada por $P = (1-c)^{V_C}$.

Com as fórmulas acima, é possível aferir o risco de um erro não ser detectado. Também é possível calcular a quantidade de desafios necessários para que uma eleição com uma determinada quantidade de eleitores e uma determinada margem de vitória apresente um risco limite de um erro não ser detectado. A seguir, são apresentados alguns casos para ilustrar os resultados obtidos.

Na Figura 2.1, o número total de eleitores V_T é fixado em 1.000.000 (um milhão) e o candidato A é o vencedor com 50,1% dos votos. Com estes números, é possível averiguar a queda do risco de não detectar um eventual erro com o aumento da quantidade de desafios. Quando a quantidade de desafios s é próxima a 1,4%, o risco de um erro não ser detectado é inferior a 10^{-6} (1 em 1 milhão).

Na Figura 2.2, o número de eleitores é novamente fixado em 1.000.000, mas, neste caso, o número de desafios é fixado em 1 a cada 1000 votos ($s = 0,1\%$). O gráfico mostra que quando o candidato A ganha com aproximadamente 51,5% dos votos, o risco de um erro não ser detectado é novamente inferior a 10^{-6}.

Por fim, na Figura 2.3, o número de eleitores é fixado em 1.000.000 e o risco de não detectar um erro na apuração é fixado em 10^{-6}. O gráfico ilustra a quantidade de desafios necessário para atingir esse risco com a variação dos votos que o candidato A recebe. É possível observar que quando o candidato A recebe 50,3% dos votos, a quantidade de desafios necessária é de 0,5%.

Para confirmar a utilidade do desafio de Benaloh em cenários reais, o mecanismo foi aplicado em três cenários no qual a diferença entre o candidato vencedor e o candidato perdedor foi pequena. Estes cenários, em ordem cronológica, são a disputa presidencial de 2º turno brasileira em 2014, a disputa presidencial norte americana no estado da Georgia

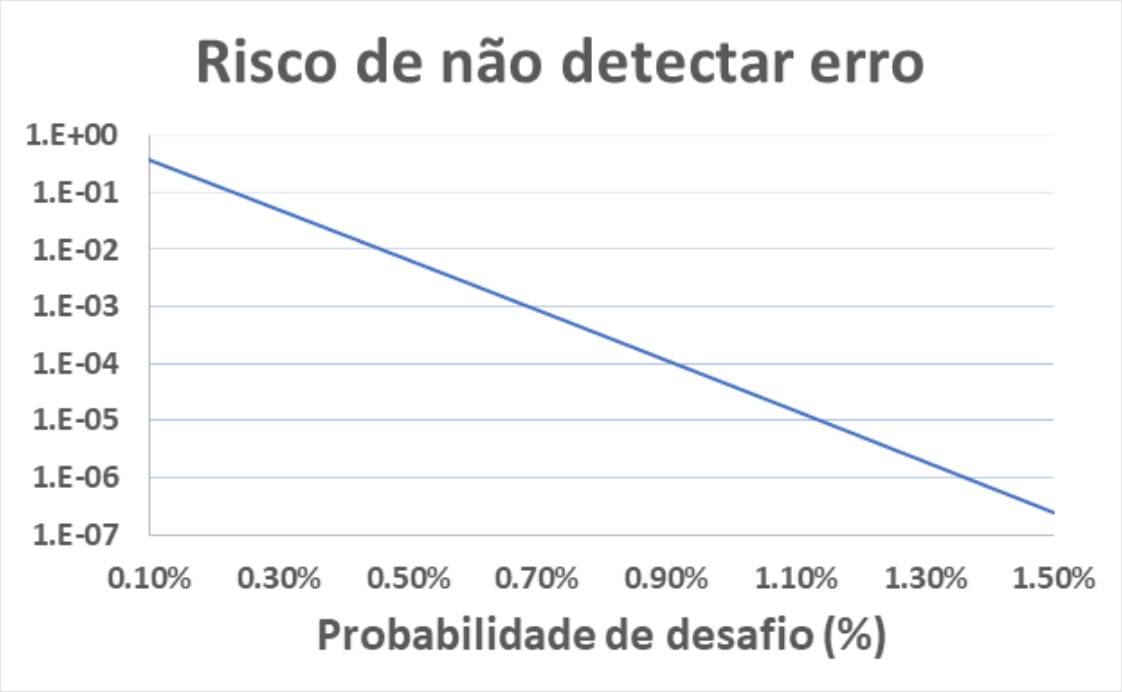


Figura 2.1: Risco de não detectar um erro no resultado eleitoral com 1.000.000 eleitores e vitória de A com 50,1%.

em 2020, e a disputa presidencial de 2º turno brasileira em 2022.

No caso das eleições brasileiras de 2014, o candidato vencedor obteve 54.501.118 votos do total de 105.542.273 votos, ou 51,64% dos votos. Para que o risco de não detectar um erro nessa eleição seja inferior a 10^{-6}, a quantidade de desafios necessária é de $s = 0,000008$, ou 1 desafio a cada 125.000 votos.

No caso das eleições americanas no estado da Georgia de 2020, o candidato vencedor obteve 2.473.633 votos do total de 4.935.487 votos, ou 50,12% dos votos. Para que o risco de não detectar um erro nessa eleição seja inferior a 10^{-6}, a quantidade de desafios necessária é de $s = 0,00235$, ou 1 desafio a cada 400 votos.

No último cenário, nas eleições brasileiras de 2022, o candidato vencedor obteve 60.345.999 votos do total de 118.552.353 votos, ou 50,90% dos votos. Para que o risco de não detectar um erro nessa eleição seja inferior a 10^{-6}, a quantidade de desafios necessária é de $s = 0,000013$, ou 1 desafio a cada 77.435 votos.

A Tabela 2.1 apresenta o resumo agregado dos resultados obtidos.

2.3.1.6. Sigilo do voto

As três propriedades fornecidas por um sistema fim-a-fim tem como objetivo garantir a **integridade** do processo de votação. Entretanto, outra propriedade de interesse em um sistema eleitoral é o **sigilo** do voto. De fato, o sigilo do voto protege o eleitor de tentativas de coerção e intimidação, garantindo sua liberdade de escolha. Infelizmente, um sistema

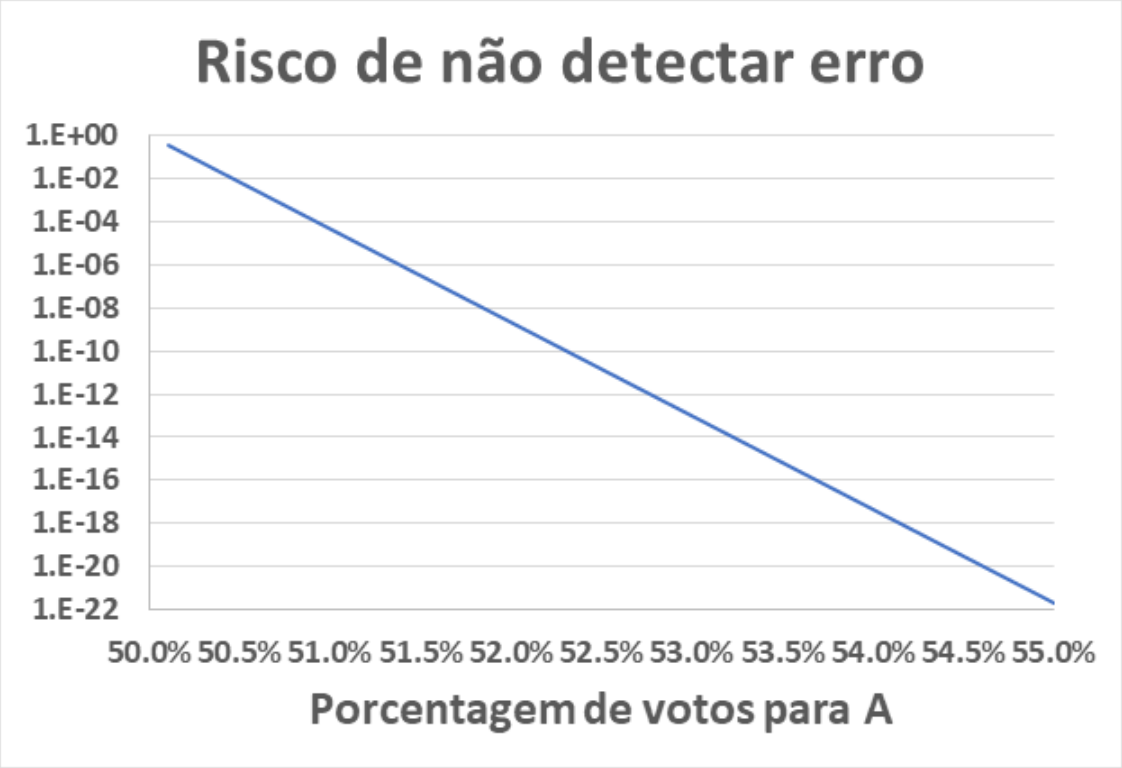


Figura 2.2: Risco de não detectar um erro no resultado eleitoral com 1.000.000 eleitores e probabilidade de desafio de 0,1%.

fim-a-fim não possuí características que melhorem a propriedade de sigilo do voto em relação a sistemas tradicionais de votação.

Em um sistema E2E, o sigilo do voto depende da confiança do eleitor na entidade ou conjunto de entidades que controlam o processo. Este fato, na pior hipótese, não é diferente do atual processo eleitoral. Quanto ao Código de Rastreio entregue a cada eleitor, este código apenas permite que o eleitor encontre seu voto **cifrado** no sistema. Logo, para que uma eventual quebra de sigilo ocorra através do Código de Rastreio, seria necessário que o esquema de cifração subjacente também seja igualmente quebrado. Além disto, um sistema fim-a-fim oferece a possibilidade de inclusão de mais entidades em alguns processos, como a guarda de chaves criptográficas. Desta maneira, é necessário o conluio de múltiplas entidades para a quebra de sigilo do voto, diluindo a confiança necessária em cada entidade individualmente.

Ademais, existe uma dificuldade inerente a proteção do sigilo do voto pelo sistema pela existência de inúmeros potenciais ataques de maneira externa e física a ele, inclusive de origem do próprio eleitor. Como exemplo de um ataque ao sigilo realizado pelo próprio eleitor, é possível citar o hábito de alguns eleitores de tirarem fotos no momento da votação, com seus votos visíveis, e a posterior postagem destas fotos em mídias sociais.

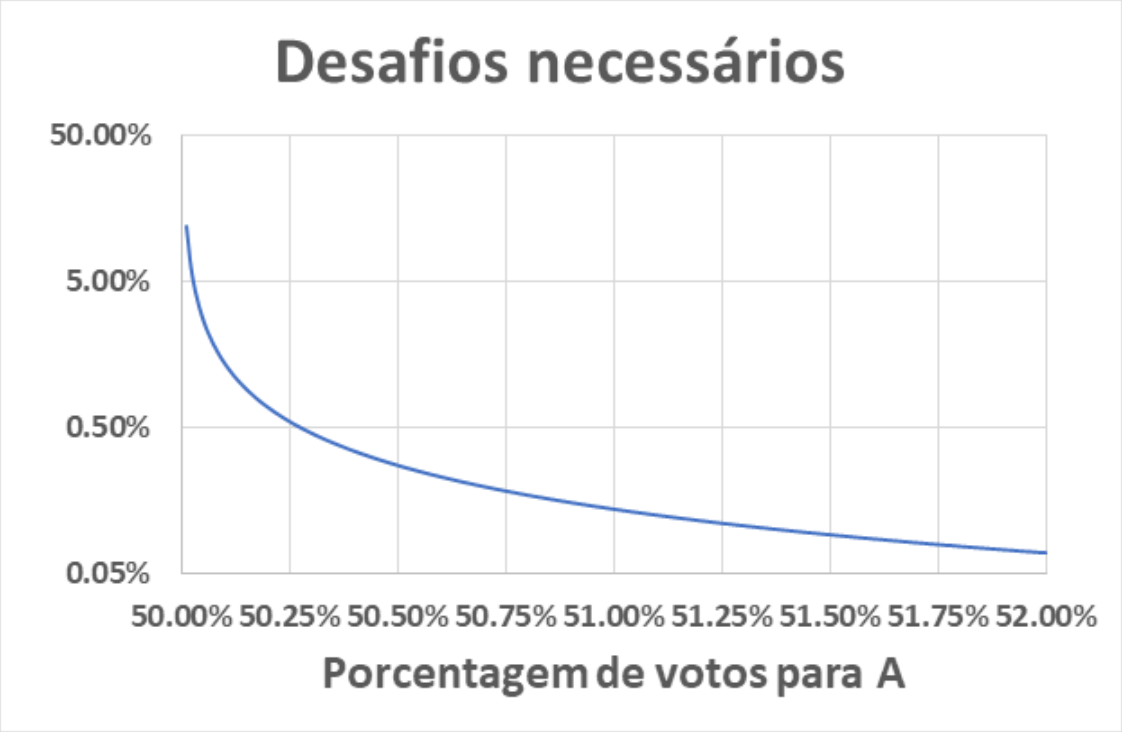


Figura 2.3: Quantidade de desafios necessária para atingir um risco de 10^{-6} com 1.000.000 de eleitores e uma porcentagem variável de votos para A.

2.3.1.7. Trilha em Papel para Auditoria Verificada pelo Eleitor - VVPAT

Um mecanismo recomendado pela literatura para ser empregado em sistemas eleitorais eletrônicos é o VVPAT, do inglês *Voter Verified Paper Audit Trail*, ou Trilha em Papel para Auditoria Verificada pelo Eleitor [Rivest 2008]. Apesar de ambos o VVPAT quanto o método fim-a-fim buscarem garantir a integridade do sistema de votação, eles atuam de maneiras diferentes, com objetivos e atores distintos.

No método VVPAT, o sistema de votação, antes de gravar o voto do eleitor, produz uma versão impressa deste voto. O eleitor, então, verifica que tanto a versão digital quanto a versão física do voto estão corretas e são iguais e, em caso positivo, permite com que o sistema grave o voto eletrônico ao mesmo tempo que deposita o voto físico (de maneira automática ou manual) em uma urna convencional. Posteriormente, o voto físico é utilizado em um processo de auditoria, chamado de Auditoria de Limitação de Risco, ou RLA (do inglês, *Risk-limiting Audit*) [Lindeman and Stark 2012].

O objetivo desta auditoria é corrigir o resultado de uma eleição caso o processo não obtenha evidências de que a apuração eletrônica esteja correta. Para isto, uma amostragem aleatória dos votos físicos é comparada com seus representantes digitais a fim de obter evidências de que uma contagem do total de votos físicos resultaria no mesmo resultado eletrônico. O "Risco Limite", presente na descrição do processo, refere-se a maior possibilidade existente de a auditoria não detectar um eventual erro no resultado da eleição, independente da origem deste erro.

Tabela 2.1: Aplicação do desafio de Benaloh em cenários reais.

Cenário	Votos para A	Total de Votos	Votos para A (%)	Desafio (Risco $< 10^{-6}$)
Eleição Presidencial Brasileira (2014)	54.501.118	105.542.273	$51,64$	$s = 0,000008$ (1/125.000 votos)
Eleição Presidencial Americana (Georgia 2020)	2.473.633	4.935.487	$50,12$	$s = 0,00235$ (1/400 votos)
Eleição Presidencial Brasileira (2022)	60.345.999	118.552.353	$50,90$	$s = 0,000013$ (1/77.435 votos)

Como esperado, quanto menor o risco que um auditor deseje obter, maior deve ser o tamanho da amostra aleatória de votos. Entretanto, a margem de vitória da eleição também influencia no tamanho da amostra, afinal, quanto menor a margem, mais evidências são necessárias, pois menor é o erro permitido. Desta maneira, é evidente que a apuração eletrônica da eleição é um processo prévio necessário ao início da auditoria.

Entretanto, como *inesperado* por grande parte da população, o tamanho da amostra aleatória é extremamente pequeno para a enorme maioria dos cenários. Mesmo eleições com margens de vitória extremamente pequenas (e.g., 1%) resultam em amostras aleatórias inferiores a 1% dos votos. Como exemplo, considerando a eleição presidencial brasileira de 2022 e utilizando a ferramenta de cálculo de número de amostras disponível em [Lindeman and Stark 2020], com risco limite de 10^{-6}, o tamanho da amostra resultante é de 1687 votos, ou aproximadamente $0,0014\%$ dos votos totais. Como curiosidade, é interessante observar que este número é bastante similar ao número de desafios necessários no processo fim-a-fim para o mesmo cenário, $0,0013\%$ dos votos totais.

Em comparação ao processo fim-a-fim, inicia-se por destacar a diferença dos atores envolvidos. No sistema E2E, o próprio eleitor é responsável por desafiar e verificar o resultado do sistema. No método VVPAT, o resultado é checado por um auditor externo, sem o envolvimento do eleitor. Desta maneira, pode-se dizer que no método VVPAT ainda é necessário, por parte do eleitor, uma confiança em algum agente, enquanto no sistema E2E o eleitor pode, em tese, executar seu próprio código para realizar a verificação do voto caso assim deseje.

Em seguida, destaca-se a diferença de objetivo entre os dois modelos. Enquanto que no VVPAT o objetivo é buscar evidências de que o resultado apurado eletronicamente está correto e corrigi-lo caso contrário, o objetivo do fim-a-fim é averiguar que o sistema está operando corretamente. Embora, em tese, os dois objetivos levem ao mesmo resultado, os modelos são submetidos a diferentes tipos de ataque. Por exemplo, não existe um mecanismo inerente ao VVPAT que impeça a substituição de um conjunto de votos eletrônicos e seus respectivos votos físicos por um conjunto falso. Já no sistema fim-a-fim, como o eleitor recebe um Código de Rastreio assinado digitalmente, ele é capaz de provar

que um voto legítimo foi trocado.

Observa-se também a facilidade de compreensão, por parte do público, entre os dois modelos. O VVPAT disponibiliza o voto às claras do eleitor para checagem, enquanto que o E2E mostra um resultado de um hash de um voto cifrado. Para um eleitor, a verificação do voto às claras é extremamente mais simples do que o cálculo matématico e garantias existentes da comparação de resultados de hashes criptográficos.

Por fim, é importante observar a questão de guarda e logística do componente impresso nos dois casos. No mecanismo VVPAT, a autoridade eleitoral é responsável por realizar a coleta, transporte e guarda dos votos físicos para posterior auditoria. Enquanto isso, no fim-a-fim, cada eleitor é responsável por seu próprio Código de Rastreio. Em votações nas quais o território e o número de eleitores é elevado, o custo inerente a logística do VVPAT pode tornar-se igualmente alto. Já o processo E2E não acarreta em custos adicionais de logística.

Como os mecanismos de VVPAT e fim-a-fim atendem a diferentes objetivos, com pontos positivos e negativos complementares, além de não serem conflitantes em termos operacionais, a implementação concomitante de ambos os modelos é possível. Ademais, caso não existam impedimentos legais ou financeiros para isto, a implementação em conjunto dos dois mecanismos é recomendada.

2.3.2. Trabalhos Relacionados de Sistemas E2E

Nesta Seção, alguns dos principais sistemas de votação fim-a-fim existentes na literatura são brevemente apresentados. Os sistemas Helios [Adida 2008] e ElectionGuard [Benaloh and Naehrig 2022] são intencionalmente omitidos, pois eles serão descritos em seções específicas.

O artigo **Prêt à Voter** [Chaum et al. 2005] descreve um esquema de votação fim-a-fim com a utilização de cédulas de votação em papel que são escaneadas. Neste esquema, cada cédula de votação é constituída de duas metades, a esquerda e a direita, e é criada individualmente. Na metade esquerda, é apresentado uma tabela com a lista de candidatos embaralhada. O embaralhamento é feito de maneira específica para cada uma das cédulas. Na metade direita, existe uma tabela em branco, alinhada com a tabela da metade esquerda, juntamente com uma linha extra, que possui um número "cebola". Este número será posteriormente utilizado para desembaralhar a cédula. Um exemplo de cédula é mostrado na Figura 2.4.

Para votar, um eleitor localiza seu candidato na lista da metade esquerda e marca o campo na metade direita (e.g., com um X). Em seguida, o eleitor separa as duas metades da cédula. A metade direita é escaneada e publicada para que o votante também possa comparar com seu recibo. O conteúdo da cédula e seu valor, o número da coluna marcada e o número cebola são, então, salvos.

Após todos os votos serem registrados, o sistema opera de maneira similar a uma rede de misturadores. Cada nó misturador, estabelecidos previamente e em uma determinada sequência, recebe o valor registrado. O nó decifra o número cebola, obtendo dois valores. O primeiro é utilizado na marcação de voto, tirando uma camada de embaralhamento. O segundo é anexado a nova marcação e enviado conjuntamente para o próximo

Basquete	
Futebol	
Natação	
Vôlei	

(a) Lista de candidatos ordenada.

Natação	X
Vôlei	
Basquete	
Futebol	
	XGJlZ2lu

(b) Exemplo de uma cédula de votação com candidatos embaralhados e voto para "Natação".

Figura 2.4: Visão do eleitor do esquema Prêt à Voter. A lista de candidatos ordenada (a) é pública e cada cédula recebe um embaralhamento diferente desta lista, com um número cebola apropriado (b).

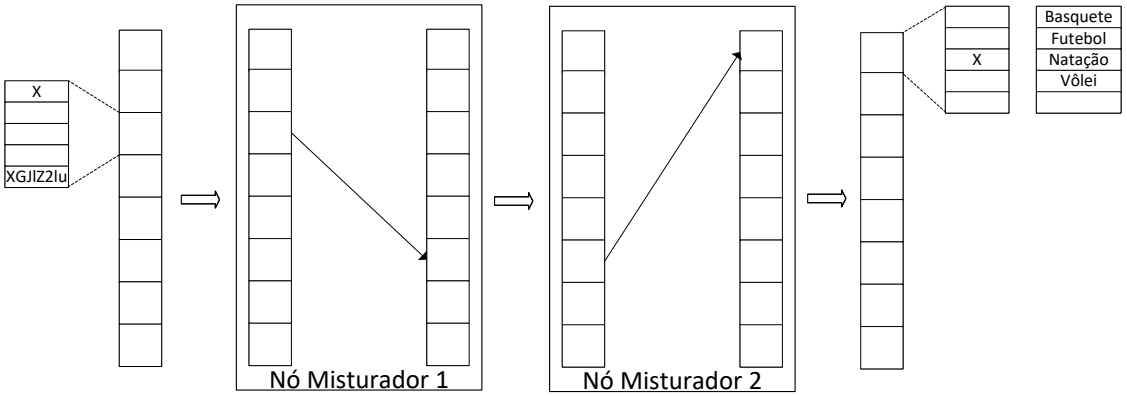


Figura 2.5: Operação simplificada do sistema Prêt à Voter. Após a coleta das cédulas de voto dos eleitores, estas são processados por uma sequência de nós misturadores, que desembaralham e alteram a ordem do conjunto de cédulas, até a produção do voto final. No exemplo, a cédula utilizada para votar em "Natação" da Figura 2.4 é desembaralhada e misturada, até a saída final, onde a marcação corresponde a lista ordenada pública de candidatos.

nó, como o novo número cebola. Adicionalmente, antes do envio do voto para o próximo nó, os votos dos eleitores são misturados, a fim de remover a associação entre a ordem de entrada e a ordem de saída do misturador. Este processo é repetido por todos os misturadores, até que a saída final do sistema corresponde a marcação do voto em uma lista ordenada dos candidatos, comum a todos os votos. Com isso, a apuração final da eleição pode ser feita.

Como todos os votos na entrada do primeiro misturador são públicos e existem processos de auditoria para garantir o correto desembaralhamento e mistura dos votos, com o comprovante de voto, o eleitor é capaz de certificar-se que seu voto foi utilizado para o cálculo do resultado final da eleição. Um resumo da operação do esquema Prêt à Voter é apresentado na Figura 2.5.

O sistema **RIES** [Hubbers et al. 2005] é um sistema de votação híbrido, que opera através da Internet ou por carta, que utiliza um mecanismo de hash associado com um de cifração. Um voto é o resultado de um hash aplicado a cifração de um identificador de

candidato.

Inicialmente, a autoridade eleitoral cria uma chave de cifração única para cada eleitor. Em seguida, esta autoridade, utilizando a chave criada para cada eleitor e para cada candidato, cifra o identificador do candidato e computa o hash deste resultado, associando o valor final com o candidato. Todos estes resultados, que constituem todos os votos possíveis, são tornados públicos. Por fim, a entidade eleitoral envia a chave de cifração para cada eleitor.

Para votar, o eleitor escolhe seu candidato e apenas cifra o identificador associado, enviando o dado cifrado a autoridade eleitoral. A autoridade eleitoral, por sua vez, publica todos os dados cifrados que recebeu e o hash destes dados. Utilizando a lista de todos os votos possíveis, publicada anteriormente, é possível associar o valor do hash calculado com o valor associado a um candidato, desta maneira contabilizando um voto para este candidato.

Como o eleitor guarda o resultado cifrado que enviou a autoridade eleitoral, ele é capaz de conferir que seu voto consta na lista final de apuração e foi utilizado. Além disto, como todos os dados cifrados são feitos públicos e qualquer pessoa é capaz de calcular o hash destes dados, a verificação do resultado da eleição pode ser feita por todos, inclusive não eleitores.

O esquema **Scantegrity II** [Chaum et al. 2008] é um sistema fim-a-fim baseado em cédulas de votação em papel com códigos de confirmação e uso de escâneres. As cédulas de votação são compostas por três partes, sendo elas o corpo principal, o voucher inferior esquerdo e o voucher inferior direito. No corpo principal, localizam-se todas as escolhas possíveis de candidato ao lado de círculos, que são preenchidos para indicar a escolha do eleitor. Estes círculos são criados com uma tinta especial e, quando marcados utilizando uma caneta específica, revelam por alguns minutos um pequeno código antes de se tornarem completamente preenchidos. O voucher inferior esquerdo contém um campo que também é criado utilizando uma tinta especial, diferente da primeira, e um espaço para anotações. O campo contém um número serial da cédula específica e o espaço para anotações pode ser utilizado pelo eleitor para copiar o código de confirmação obtido ao realizar a escolha de candidato no corpo principal. Por fim, o voucher inferior direito é composto por um campo, criado com a mesma tinta especial do voucher esquero, que contém um outro número serial associado a cédula, diferente do anterior. Um exemplo de cédula marcada do Scantegrity é apresentado na Figura 2.6.

Para votar, um eleitor recebe uma cédula do mesário e dirige-se a cabine de votação. Na cabine, o eleitor utiliza uma caneta reagente a tinta especial dos círculos do corpo principal para realizar sua escolha. Ao retornar da cabine, a cédula é inserida em um escâner, que verifica se ela foi corretamente preenchida (i.e., apenas uma escolha foi realizada). Em caso negativo, a cédula é descartada e o eleitor deve realizar a votação novamente. No descarte, é escolhido arbitrariamente o voucher esquerdo ou direito, que é retido pelo mesário. Em caso positivo, o voto é salvo, o corpo principal é depositado em uma urna e os vouchers inferiores são fornecidos ao eleitor. Em seguida, o eleitor apresenta ambos os vouchers ao mesário, que utilizando uma caneta que reage a tinta específica dos vouchers, revela ambos os números seriais associados a cédula. Por fim, o eleitor com posse de ambos os vouchers com seus números seriais revelados e sua marca-

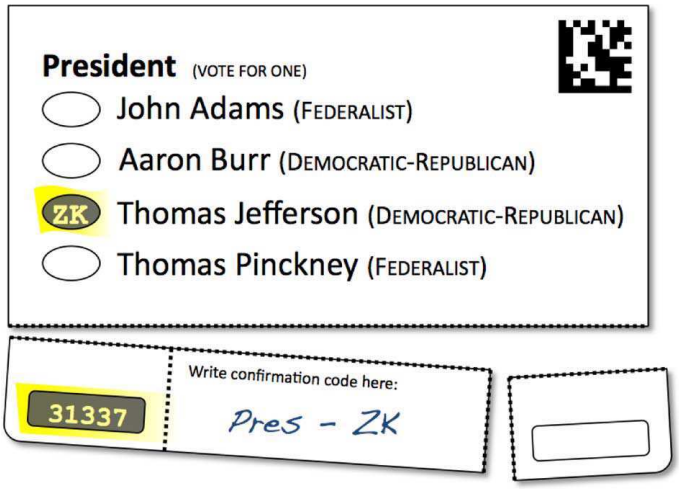


Figura 2.6: Exemplo de cédula do esquema Scantegrity II, com corpo principal e vouchers inferiores esquerdo e direito. O voto contido na cédula corresponde ao candidato Thomas Jefferson. Fonte: [Chaum et al. 2008]

ção de código de confirmação, conclui o processo de votação.

Após o encerramento da eleição, o eleitor é capaz de acessar um site específico que, ao ser informado de um dos números seriais associados a cédula, retorna o código de confirmação da escolha do eleitor. Como os códigos de confirmação são gerados de maneira criptográfica e associados previamente a cada cédula e a cada candidato, o eleitor pode confirmar que sua escolha foi corretamente capturada e seu voto utilizado para a apuração. Caso o eleitor obtenha um código de confirmação diferente do anotado, um processo de auditoria é iniciado para detectar a causa do problema. Observa-se que o caso no qual o eleitor cometa um erro, intencional ou não, é facilmente detectado. Isto é possível pois o sistema é capaz de produzir todos os códigos de confirmação de uma cédula e uma auditoria é capaz de verificar se o eleitor produziu um código válido ou não para a cédula específica, sendo a chance de acertar ao acaso um código válido parametrizável.

O sistema fim-a-fim **PunchScan** [Popoveniuc and Hosp 2010] é um esquema baseado em cédulas de papel. Neste esquema, a cédula de votação é composta por duas páginas, que são sobrepostas. Na página superior, são apresentados ao eleitor os candidatos e seus códigos de identificação, que são embaralhados para cada cédula, sendo a associação conhecida apenas pela autoridade eleitoral. Na página inferior, são apresentados apenas os códigos de identificação, em ordem aleatória, sendo esta ordem também apenas conhecida pela autoridade eleitoral. Adicionalmente, a página superior apresenta orifícios, que são alinhados com os códigos de identificação da página inferior. Desta maneira, quando o eleitor recebe a cédula de votação, ele é capaz de ver os candidatos e seus códigos na página superior e os códigos na página inferior, através dos orifícios. Um exemplo de cédula de votação é ilustrado na Figura 2.7.

Para votar, o eleitor utiliza um marcador cuja dimensão é superior ao tamanho do orifício da página superior. Consequentemente, ao marcar uma das opções da página inferior, a borda do orifício da página superior também é marcada. Após realizar a marcação, o eleitor separa as duas páginas da cédula, escaneando uma delas e destruindo a

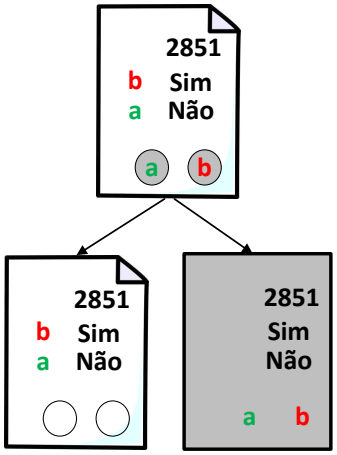


Figura 2.7: Cédula de votação do sistema Punchscan. Em cima, a visão do eleitor, com as duas páginas sobrepostas. Embaixo, à esquerda a página superior, e à direita a página inferior.

outra. A decisão de qual página é depositada é tomada antes do eleitor receber a cédula. Adicionalmente, o eleitor retém a página escaneada.

Para realizar a apuração, a autoridade eleitoral recebe as diversas páginas escaneadas e realiza a associação entre a marcação em cada página e um candidato. Isto é possível pois, como mencionado anteriormente, a autoridade sabe a associação entre candidato e identificação para cada cédula e a ordem que estas identificações são apresentadas na cédula. Por sua vez, o eleitor, com a página retida, é capaz de verificar, uma vez que a autoridade eleitoral publique as páginas recebidas, que a marcação em sua página corresponde a marcação da página recebida pela autoridade.

Para concluir esta Seção, observa-se que o artigo [Kelsey et al. 2010] cita ataques a alguns dos esquemas previamente apresentados. Por questões de espaço e por não ser o enfoque deste minicurso, estes ataques não serão comentados, deixando a análise a cargo do leitor.

2.4. Sistema E2E baseado em Redes de Misturadores

Esta Seção apresenta um sistema fim-a-fim baseado em redes de misturadores. São discutidos as técnicas utilizadas no esquema para atingir os objetivos de integridade e verificabilidade, baseando-se nos blocos criptográficos apresentados na Seção 2.2. Por fim, será apresentada a instalação da biblioteca Verificatum para a operação de uma rede de misturadores.

2.4.1. Redes de Misturadores

Redes de misturadores formam um conjunto de protocolos que garantem a comunicação segura entre duas partes [Chaum 1981, Sampigethaya and Poovendran 2006]. Originalmente proposta para uso com protocolos de roteamento, esta tecnologia permite que mensagens trafegadas entre um remetente e um destinatário não possam ser rastreadas por

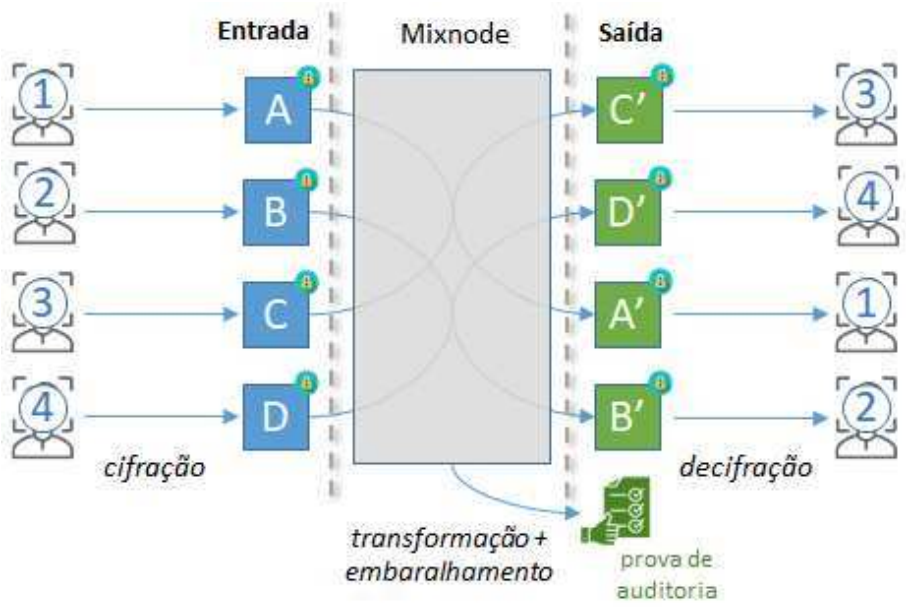


Figura 2.8: Operação de um “mixnode” ou nó misturador.

observadores na rede. A ideia consiste essencialmente no seguinte. Primeiramente, todas as mensagens dos usuários devem ser enviadas cifradas pela rede. Em alguns pontos no trajeto dessas mensagens, são incluídos então servidores proxies adicionais, chamados de “mixnodes” ou “mixservers” em inglês, ou de nós misturadores em português. Esses servidores aguardam o recebimento de um determinado volume de mensagem de diferentes usuários, e realizam algumas operações sobre essas mensagens para garantir privacidade e integridade dos dados trafegados. O resultado é uma espécie de “embaralhamento” das mensagens enviadas, não sendo possível a um observador externo ligar uma mensagem saindo do nó misturador à mensagem de origem correspondente: apenas o nó misturador conhece essa relação.

De forma geral, cada nó misturador precisa realizar três operações sobre os textos cifrados para garantir essas propriedades de segurança (vide Figura 2.8): transformação, embaralhamento e prova de auditoria. A transformação é feita para que os textos cifrados na entrada de cada nó misturador sejam modificados, mas ainda de forma que sua decifração na saída da rede de misturadores leve ao texto às claras original; o objetivo é impedir que um observador reconheça que uma mensagem de saída se relacione a uma mensagem de entrada simplesmente verificando a igualdade entre os textos cifrados em questão. Seguindo o mesmo princípio, o embaralhamento garante que também não haja relação de ordem entre as mensagens transformadas, ou seja, esconde-se a ordem entre entrada e saída. Já o mecanismo de prova de auditoria é utilizado para detectar casos em que um nó misturador malicioso tenta modificar as mensagens trafegadas (e.g., substituindo os textos cifrados correspondentes, em vez de realizar a operação de transformação que preserva o texto às claras original); em outras palavras, as provas de auditoria geradas devem ser válidas apenas para nós misturadores honestos, que não tenham tentado fazer esse tipo de modificação.

Assim, a segurança das redes de misturadores é projetada de tal forma que, mesmo

que o atacante entre em conluio com a maioria dos nós misturadores, a privacidade das mensagens é mantida se um número pequeno de nós misturadores se mantiver honesto (em alguns casos, apenas um mixnode honesto é o suficiente). Além disso, a integridade das mensagens pode facilmente ser confirmada, pois a modificação de qualquer mensagem pode ser detectada por qualquer entidade que deseje auditar o sistema.

2.4.1.1. Privacidade

Para garantir a privacidade das mensagens, os nós misturadores são capazes de transformar as mensagens cifradas de diversos usuários sem precisar revelar essas mensagens, e a ligação entre mensagens é perdida realizando um embaralhamento. A transformação da mensagem cifrada é usualmente feita por decifrações sucessivas ou por recifração dos textos cifrados.

Nas redes de misturadores de decifração [Chaum 1981], os usuários devem selecionar um conjunto de nós misturadores e, então, realizar a cifração sucessiva da mensagem utilizando as chaves de cada um deles. Quando um nó misturador receber a mensagem, ele deve decifrá-la utilizando a sua chave, embaralhar (permutar) as diferentes mensagens entre si, e só então enviar o novo conjunto de mensagens para o próximo mixnode. O último nó misturador envia a mensagem para o destinatário correto, sendo este o único capaz de decifrar a última camada da mensagem.

Formalmente, para cifrar uma mensagem m com a aplicação da cifração $E_{sk}(\cdot)$ utilizando chave de cifração pk, realiza-se a composição de cifrações:

$$C_n = E_{pk_n} \circ E_{pk_{(n-1)}} \circ \cdots \circ E_{pk_1}(m)$$

Para decifrar, cada nó misturador aplica a decifração $D_{sk}(\cdot)$ no texto cifrado C de forma iterativa, obtendo textos cifrados parciais $C_{i-1} = D_{sk_i}(C_i)$. Desta forma, a mensagem original é obtida após a composição das diversas decifrações:

$$m = C_0 = D_{sk_1} \circ D_{sk_2} \circ \cdots \circ D_{sk_n}(C_n)$$

Já redes de misturadores de recifração [Park et al. 1994] são realizadas utilizando apenas a chave pública do destinatário com uma cifra que permite recifração (e.g., a cifra ElGamal, apresentada na Seção 2.2). Quando um nó misturador recebe mensagens cifradas, ele utiliza a chave pública de cifração do destinatário para realeatorizar os textos cifrados, o que resulta em um outro texto cifrado que, ao ser decifrado, resultará na mensagem original. Em seguida, os textos recifrados são embaralhados e enviados para o próximo nó misturador. Como apenas a chave pública do destinatário é utilizada, nenhum nó misturador é capaz de decifrar as mensagens trafegadas: apenas o destinatário será capaz de fazê-lo.

Formalmente, a mensagem m é cifrada por meio da função de cifração $E_{pk}(\cdot)$ com a chave pública pk, utilizando o nonce r, apenas uma vez pelo usuário. O texto cifrado resultante, $C = E_{pk}(m;r)$, é enviado para os nós misturadores que aplicam a recifração $R_{pk}(\cdot)$, com nonce r', gerando o novo texto cifrado $C' = R_{pk}(C;r')$. Ao final da transmissão, após receber o texto cifrado $C^{(n)}$, o destinatário decifra a mensagem aplicando

a decifração $D_{sk}(\cdot)$, resultando em $m = D_{sk}(C^{(n)})$. Este processo foi detalhado na Seção 2.2.

Como mencionado anteriormente, independentemente do método de transformação que a rede de misturadores utiliza, ela precisa ser acoplada a um mecanismo de embaralhamento, misturando a ordem das diferentes mensagens para garantir privacidade. Mais precisamente, a transformação sobre os textos cifrados faz com que seja computacionalmente inviável diferenciar se a saída do nó misturador reflete a mesma mensagem recebida na entrada. Obviamente, isso assume que o nó misturador receba em sua entrada mais do que um elemento, pois caso contrário seria imediato identificar que a única mensagem de entrada e de saída pertencem ao mesmo usuário. Com diversas mensagens transformadas por iteração, o observador só conseguiria identificar quais mensagens pertencem ao mesmo usuário pela sua ordem na entrada e na saída. Portanto, ao embaralhar as mensagens antes de criar a lista de saída, esta relação deixa de existir. Logo, apenas o próprio nó misturador que conhece o embaralhamento usado seria capaz de fazer esta identificação.

2.4.1.2. Integridade

Para garantir integridade, as redes de misturadores podem utilizar mecanismos de desafio ou de provas de conhecimento. Redes de misturadores que utilizam *desafios*, por exemplo, permitem que usuários mandem mensagens marcadas de tal forma que, após serem reveladas, exigem que os nós misturadores revelem o trajeto que sua mensagem fez no sistema. Quanto mais desafios forem respondidos pela rede de misturadores, maior o grau de confiança que o processamento das mensagens não-reveladas também tenha sido feita corretamente. Já redes de misturadores que utilizam *provas de conhecimento* permitem a criação de um artefato criptográfico que pode ser verificado para garantir a corretude dos embaralhamentos, decifrações e recifrações. É importante notar que um nó misturador desonesto não seria capaz de criar um artefato caso ele tenha adulterado as mensagens, já que a verificação dessas provas resultaria em falha.

2.4.1.3. Comparação entre os tipos de redes de misturadores

A principal desvantagem de redes de misturadores baseadas em decifração é o fato de que deve haver gerenciamento de chaves de todos os nós misturadores da rede. Não somente a chave de todos os nós misturadores deve ser sincronizada no início do envio da mensagem, como o processo deve ser reiniciado caso algum nó misturador seja desconectado em operação. Já uma rede de misturadores de recifração depende apenas de uma chave pública global do sistema, e nós misturadores intermediários que falharem podem ser removidos facilmente das operações.

Com relação a integridade, a maior desvantagem do uso de desafios provém do fato de que desafios são mensagens adicionais que precisam ser enviadas em tempo real. Isso resulta no aumento da banda usada pelo sistema, e o número de desafios pode variar a cada iteração do uso. Além disso, como os desafios são realizados por usuários do sistema, o aumento de confiança só se dá por desafios confiáveis, isto é, desafios enviados

pelo próprio usuário, ou por outro usuário da sua rede de confiança. Usar provas de conhecimento, por outro lado, permite a verificação matemática de que a execução completa daquela iteração foi correta. Além disso, caso sejam utilizadas provas não-interativas, essa verificação não precisa ser feita em tempo real, mas é realizada por qualquer entidade, participante ou não do sistema.

Dadas as vantagens e desvantagens dos esquemas de redes de misturadores existentes, optou-se neste minicurso pelo uso de uma rede de misturadores de recifração (especificamente, baseada no criptosistema ElGamal), com o uso de provas de conhecimento não-interativas. As provas de conhecimento não interativas utilizadas são as provas de Wikström-Terelius, apresentadas na Seção 2.2 Nota-se que esta escolha foi voltada à eficiência, mas que o esquema sugerido pode ser substituído por outros.

2.4.2. Redes de misturadores em sistemas de votação

As propriedades de segurança voltadas à privacidade e integridade dos votos permitiu que surgissem propostas de sistemas de votação utilizando estas ferramentas [Jakobsson et al. 2002]. Isso pode ser observado em esquemas de votação online (e.g. Helios [Adida 2008]) ou esquemas presenciais (e.g., Prêt-à-Voter [Chaum et al. 2005]), inclusive em votações públicas na Noruega e Estônia, demonstrando a viabilidade deste tipo de construção.

Propõe-se aqui a aplicação de redes de misturadores em dois cenários distintos. O primeiro, chamado de **tradicional**, é semelhante a outras soluções de votação online como na primeira versão do sistema Helios. Ele refere-se ao uso de redes de misturadores de forma distribuída para garantir sigilo, inclusive em relação à autoridade eleitoral. O segundo, chamado de **modelo de auditoria**, é proposto com o uso de redes de misturadores para garantir a integridade dos votos dos eleitores de maneira mais centralizada e como ferramenta de auditoria da totalização dos votos.

No que se segue, o sistema de totalização é tratado como uma única entidade da autoridade eleitoral, sendo responsável pela decifração dos votos dos eleitores e sua conseguinte totalização. No entanto, é importante notar que este sistema pode não ser unitário, mas pode ser composto por diversos guardiões. Neste caso, cada guardião possui uma parte da chave privada de decifração relacionada a uma única chave pública (e.g., utilizando a versão limiar do criptosistema ElGamal descrito na Seção 2.2.3.4), e seria necessária a colaboração de um subconjunto destes guardiões para realizar a decifração total dos votos. Esta propriedade aumenta a confiança pois uma única autoridade eleitoral não seria capaz de quebrar o sigilo dos votos. Para isso, ela precisaria entrar em conluio com os outros guardiões do sistema.

2.4.2.1. Cenário tradicional

A arquitetura deste cenário é bastante semelhante ao cenário do sistema Helios, porém adaptado ao voto presencial. Para isso, tornam-se necessários os seguintes componentes: uma urna, como plataforma de votação; um conjunto de nós misturadores, onde pelo menos um não pertença à autoridade eleitoral; e o sistema de totalização dos votos.

Como preparação para a eleição, o sistema de totalização é responsável por criar

um par de chaves pública-privada para ser utilizada pela urna e pelos nós misturadores para (re-)cifração dos votos. A chave pública de cifração deve ser carregada previamente em todas as urnas e enviada para todos os nós misturadores, e apenas o sistema de totalização deve conhecer a chave privada de decifração.

No processo de votação, a urna funciona como plataforma de votação e é responsável por realizar corretamente a cifração dos votos dos eleitores. Quando o eleitor confirmar suas escolhas, a urna deve emitir o Código de Rastreio, como apresentado na Seção 2.3.1.4. Caso o eleitor deseje depositar este voto, o Código de Rastreio deve ser assinado e poderá ser utilizado posteriormente como material de auditoria da integridade dos votos contabilizados.

Neste momento, como o voto não é criado por dispositivo do próprio eleitor, este pode não ter a confiança de que a urna está realizando a operação de forma correta, ou seja, ele pode desconfiar que o comprovante de votação está apresentando um valor diferente do inserido na urna. Por este motivo, o eleitor pode desafiar a urna por meio do desafio de Benaloh como descrito na Seção 2.3.1.5.

Ao final do processo de votação, inicia-se o procedimento das redes de misturadores. Os votos cifrados depositados na urna são então enviados para o primeiro nó misturador da rede. Este nó pode utilizar a carga de votos de uma urna, ou aguardar o recebimento da carga de diversas urnas para realizar as operações em conjunto. Após o recebimento, a rede de misturadores faz sua operação comum: transforma e embaralha os textos cifrados, cria provas de que as operações foram feitas corretamente, e envia para o próximo nó da rede, utilizando as provas de conhecimento descritas na Seção 2.2. O último nó, por fim, envia os votos (re-)cifrados para o sistema de totalização, que deve ser capaz de decifrar os votos de maneira verificável. Os votos decifrados, agora sem qualquer vínculo com os respectivos eleitores, podem então ser totalizados.

Para a auditoria, todos os nós misturadores devem deixar públicas as tabelas de entrada e saída de votos cifrados, e suas correspondentes provas de corretude. O sistema de totalização também deve deixar públicos os votos cifrados de entrada e os decifrados de saída, juntamente com provas de correta decifração. Eleitores que desejem auditar os seus votos podem, neste momento, verificar se o voto cifrado do seu Código de Rastreio está presente nas tabelas de entradas do primeiro nó misturador, o que garante que ele trafegou por todo o sistema e entrou na totalização.

2.4.2.2. Cenário de auditoria

Ao contrário do cenário tradicional, a arquitetura deste sistema é voltada para dar poder de auditoria aos votos em uma urna eletrônica. Considera-se que esta é suficiente para dar privacidade aos votos dos eleitores contra atacantes externos, enquanto o uso de guardiões mantém a privacidade em relação à autoridade eleitoral (por prevenir que esta última decifre os comprovantes dos eleitores). Desta forma, o sistema, agora mais simples, conta com apenas dois componentes: a urna eletrônica, que faz o papel da plataforma de votação e também do único nó misturador no sistema; e a entidade de totalização.

O processo de preparação da eleição e o processo de votação ocorrem como no

cenário tradicional. O sistema de totalização gera o par de chaves pública/privada, e então carrega a chave pública de cifração na urna; já a chave privada de decifração é mantida em segredo. Os eleitores têm seus votos cifrados pela urna e seus Códigos de Rastreio assinados, podendo realizar o desafio nas urnas como no Cenário Tradicional.

A diferença ocorre ao final do processo de votação. Nesta etapa, apenas a urna realiza a função da rede de misturadores, transformando e embaralhando os votos, e criando as provas de corretude desta operação. Os votos cifrados originais e os recifrados são então enviados juntamente com as provas para o sistema de totalização. Os votos passam pelo processo de decifração verificável como no cenário tradicional, e as tabelas de votos cifrados e decifrados, bem como as provas de auditoria, são publicadas para verificação por potenciais interessados.

Cabe notar que, nesse cenário em que a urna é o único nó misturador no sistema, preserva-se essencialmente a mesma confidencialidade dos votos obtida com uma urna eletrônica tradicional brasileira. Para entender melhor essa observação, considere um cenário em que a urna eletrônica atual seja subvertida, passando então a associar os eleitores a seus respectivos votos às claras; isso poderia ser feito, por exemplo, por meio da criação de uma tabela associando o título do eleitor a seu voto, ou simplesmente guardando a ordem dos votos lançados (deixando para o atacante a tarefa de identificar a ordem dos eleitores na fila). Se isso for feito, a adição de nós misturadores em pontos seguintes do sistema, como previsto no cenário tradicional, não seria capaz de garantir o sigilo dos votos. Logo, a urna eletrônica já é um ponto crítico para garantir o sigilo das escolhas do eleitor, devendo ser protegida contra tentativas de subversão. Assumindo que essa proteção seja efetiva, a atuação da urna como o primeiro (e único) nó misturador já garante o sigilo dos votos, mesmo que não haja nós misturadores adicionais posteriormente.

Isso posto, esta abordagem parece ser uma escolha natural para o contexto de votação brasileiro, em que o dispositivo de entrada não é controlado pelo usuário (contrariamente ao cenário tradicional, em que se assume que as mensagens de entrada da rede de misturadores é gerada por um dispositivo de confiança do usuário). Como benefício adicional, pode-se considerar a possibilidade de combinar os resultados de mais de uma urna com um nó misturador adicional, com o objetivo de reduzir a possibilidade de inferir votos a partir dos resultados parciais individuais de uma única urna. Por exemplo, quando o resultado parcial de uma urna indica que todos os votos foram lançados para um mesmo candidato X, o sigilo de cada voto individual é naturalmente perdido: afinal, tem-se a certeza que cada eleitor daquela urna votou em X. Já se o resultado parcial individual dessa urna for omitido, mas apresentado apenas após sua combinação com o resultado de uma outra urna com os mesmos cargos, reduz-se a probabilidade de aparecimento de parciais nos quais um candidato é (quase) unanimidade. Entretanto, como a apresentação de resultados parciais individuais de cada urna é uma tradição do sistema eleitoral brasileiro, e considerado por muitos um mecanismo de auditoria relevante, esta possibilidade de omitir alguns resultados em favor de se apresentar apenas a combinação dessas parciais por meio de nós misturadores adicionais não é aqui explorada mais a fundo.

2.4.3. Verificatum

O Verificatum é uma implementação open-source de redes de misturadores de recifração usando cifra ElGamal. Esta biblioteca já foi utilizada em sistemas eleitorais online em Israel (Wombat), Espanha (Agora Voting), Noruega (Scytl), e Estônia. Ele foi escrito na linguagem de programação Java, com trechos de código em C por questões de eficiência. Além da funcionalidade básica de embaralhamento com recifração de textos cifrados, ele também conta com provas de conhecimento-zero interativas e não-interativas, decifração parcial verificável de textos cifrados, e capacidade de pré-processamento para aceleração de operações em tempo real.

As primitivas criptográficas de chave pública utilizadas pelo Verificatum partem do uso da biblioteca GMP (Gnu Multiple Precision), de onde partem implementações eficientes de aritmética de corpos finitos e curvas elípticas para criptografia. Essas operações foram expandidas para que operações vetoriais pudessem ser executadas simultaneamente, e utilizam a notação multiplicativa (comum para corpos finitos) pela transparência das primitivas utilizadas.

A verificabilidade das operações provém do uso de provas de conhecimento-zero, que podem ser implementadas de forma interativa ou então de forma não-interativa, utilizando a heurística de Fiat-Shamir. Essas provas de conhecimento são utilizadas para provar três argumentos: (1) que os textos-cifrados foram embaralhados e recifrados, (2) que as mensagens foram decifradas corretamente, e (3) que o pré-processamento foi executado corretamente. Assim, tais provas são as garantias matemáticas de que a operação da rede de misturadores foi feita de maneira honesta, garantindo a integridade das mensagens. Apesar do Verificatum implementar os próprios verificadores, também existem guias para implementação de verificadores por terceiros que atendam aos requisitos necessários para validar as provas criadas.

2.4.3.1. Instalação

O projeto Verificatum é composto por 6 bibliotecas:

- GMPMEE: extensão da biblioteca GMP para operações simultâneas e de base fixa mais velozes em corpos finitos.
- VEC: biblioteca de curvas elípticas com operações simultâneas e base fixa mais velozes. Possui trechos de código do OpenSSL para otimização de operações em algumas curvas padronizadas.
- VMGJ: interface das bibliotecas de baixo nível GMP e GMPMEE para aplicações em Java.
- VECJ: interface da biblioteca de baixo nível VEC para aplicações em Java.
- VCR: biblioteca contendo as principais rotinas modularizadas necessárias para a execução de uma rede de misturadores baseada na cifra ElGamal.
- VMN: implementação dos protocolos executados na rede de misturadores.

Estas bibliotecas podem ser encontradas no repositório do github do projeto (`https://github.com/verificatum`), e as versões utilizadas no teste foram clonadas das mais recentes no dia 1 de agosto de 2022. Eles foram realizados na versão 22.10 do Ubuntu e os seguintes pacotes foram instalados do repositório padrão como requisitos do projeto: libgmp3-dev, automake, libtool, e default-jdk.

Para a instalação de cada uma das bibliotecas, os seguintes comandos devem ser usados:

```
# autoreconf -vfi
# make -f Makefile.build
# ./configure
# make
# sudo make install
```

Algumas bibliotecas também exigem comandos adicionais após sua instalação:

- VMGJ: acrescentar a biblioteca no classpath do Java:

```
# export CLASSPATH=/usr/local/share/java/verificatum-vmgj-1.2.2.
    jar:${CLASSPATH}
# export LD_LIBRARY_PATH=/usr/local/lib:${LD_LIBRARY_PATH}
```

- VECJ: acrescentar a biblioteca no classpath do Java:

```
# export CLASSPATH=/usr/local/share/java/verificatum-vecj-2.1.5.
    jar:${CLASSPATH}
```

- VCR: ativar o uso das bibliotecas VMGJ e VECJ, e inicialização da fonte de aleatoriedade:

```
# ./configure --enable-vmgj --enable-vecj
# vog -rndinit RandomDevice /dev/urandom
```

Adicionalmente, as bibliotecas VMGJ, VECJ, VCR e VMN possuem testes unitários para verificação da instalação pelo comando:

```
# make check
```

2.4.3.2. Avaliação de eficiência

Para avaliar a eficiência dos métodos do Verificatum, foram analisadas suas três principais funcionalidades:

- *Shuffle*: embaralhamento e recifração dos textos cifrados, juntamente com a criação das provas de conhecimento-zero das operações.
- *Decrypt*: decifração dos textos cifrados, com a criação das provas de conhecimento-zero que garantem a correta decifração.

- *Mixing*: operação conjunta de embaralhamento, recifração e decifração dos textos cifrados, quando executados por apenas um nó.

Os testes foram feitos em um hardware com as seguintes características:

- Processador: Intel Atom Elkhart Lake Processor, J6412/X6413E
- Memória: 2 x 8GB SODIMM DDR4 3200Mhz
- Disco: SSD Crucial P2 500 GB 3D NAND NVMe PCIe M.2 (Sequential Read - 2300 MB/s, Sequential Write - 940 MB/s)
- Sistema operacional: Ubuntu 22.04.1 LTS

A biblioteca foi configurada para utilizar a cifra ElGamal baseada em curvas elípticas. Para garantir o nível de segurança de 128 bits, a curva selecionada foi a curva P-256, presente na implementação do Verificatum, e contém trecho de código em assembly da biblioteca openssl para aceleração das operações.

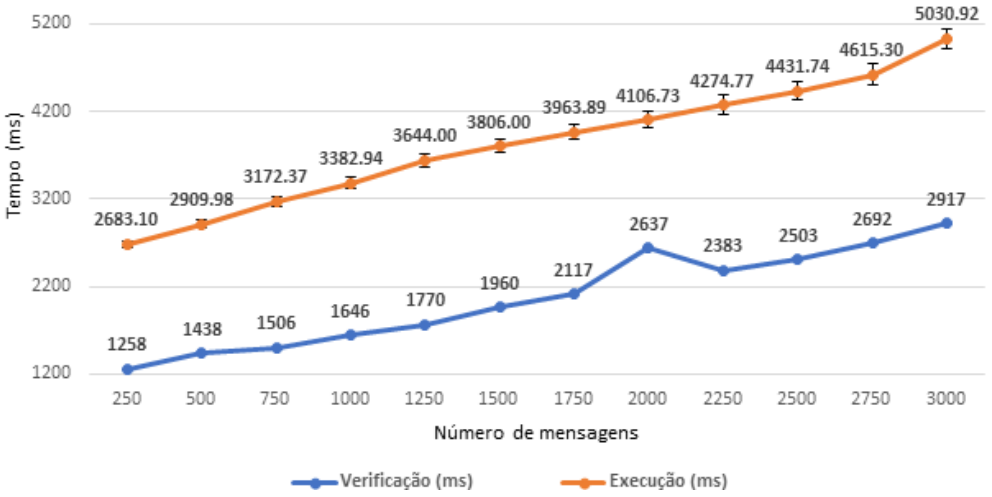
Para cada uma das funcionalidades, foram realizadas 1000 execuções independentes, com chaves geradas aleatoriamente a cada chamada para avaliar variações no tempo de execução e tamanho das provas de conhecimento-zero. Em cada execução, variou-se a dimensão dos vetores de textos cifrados entre 250 e 3000 mensagens (com passo de 250). Estes valores foram definidos considerando cenários próximos ao atualdo sistema brasileiro, em que cada urna atenda até 500 eleitores em uma eleição de até 6 cargos e cada cargo gera uma mensagem. Cabe notar, entretanto, que a separação de candidatos por cargo permite reduzir esse cenário em que 3000 mensagens são embaralhadas para um cenário em que 6 grupos distintos de 500 mensagens são embaralhados independentemente.

Para exemplificar o pior caso em questão, com 3000 mensagens, a Tabela 2.2 apresenta esse cenário para cada método executado pelo nó misturador. Pode-se observar que os tempos de execução e verificação do método "mixing" é um pouco menor que a soma dos tempos de "shuffle + decrypt". Além disso, nota-se que as provas de conhecimento-zero com custo predominante são aquelas do embaralhamento.

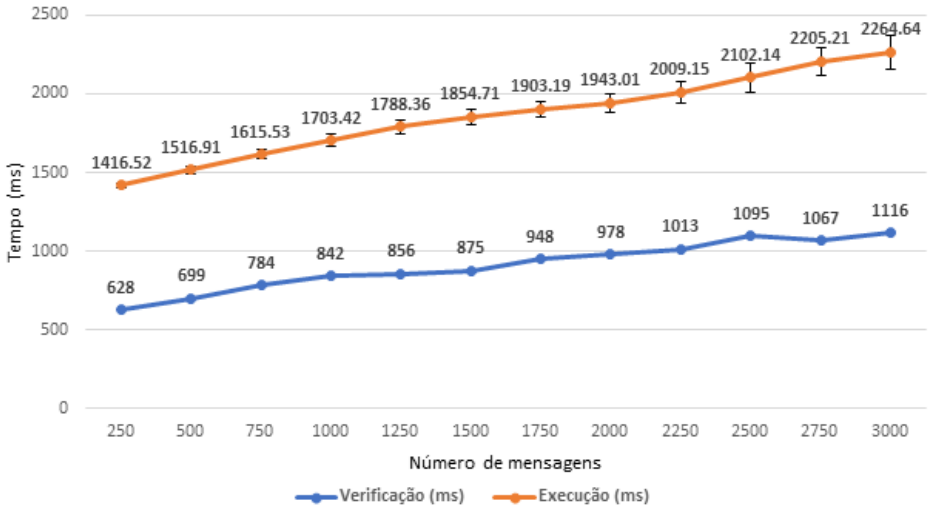
Tabela 2.2: Tempos de execução e verificação e tamanho das provas para 3000 mensagens

Método	Tempo (ms)		Tamanho (KB) Provas
	Execução	Verificação	
Shuffle	5030,92	2917	2359
Decript	2264,64	1116	949,72
Mixing	5935,27	3338	2359

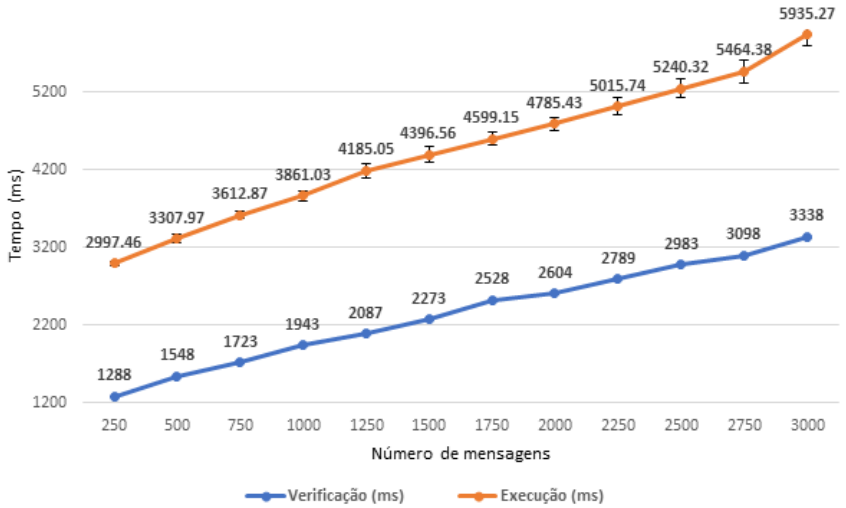
Para entender melhor como esses custos escalam, os gráficos da Figura 2.9 a seguir apresentam a média dos tempos de execução e verificação de cada método com a variação do número de textos cifrados. Já os gráficos da Figura 2.10 apresentam o tamanho das provas de conhecimento-zero que devem ser publicadas para realização da auditoria.



(a) Tempo de execução da operação Shuffle

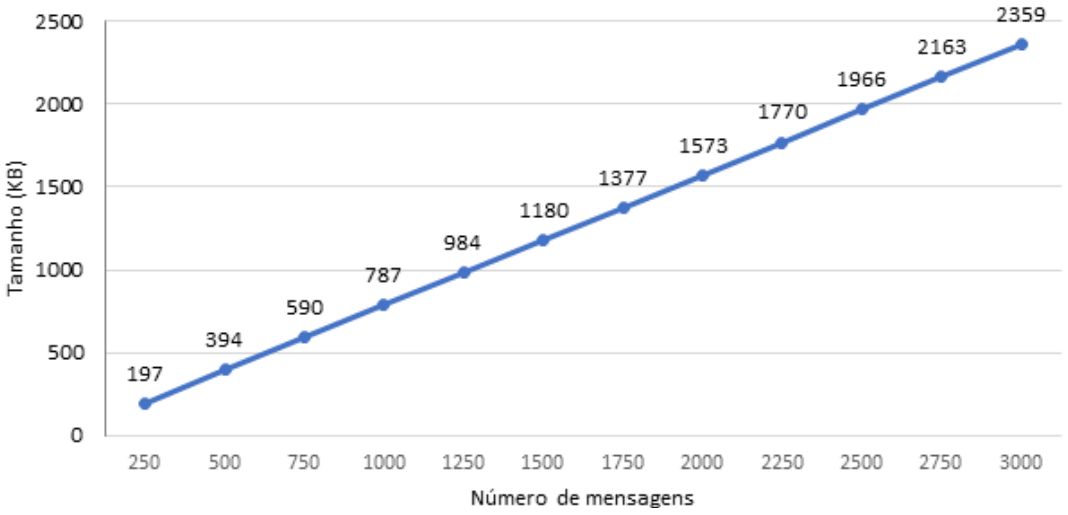


(b) Tempo de execução da operação Decrypt

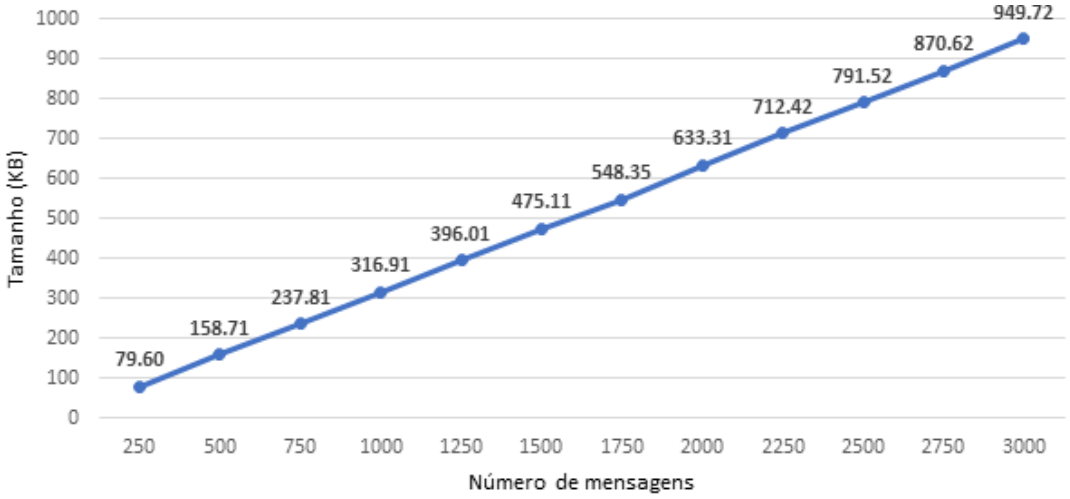


(c) Tempo de execução da operação Mixing

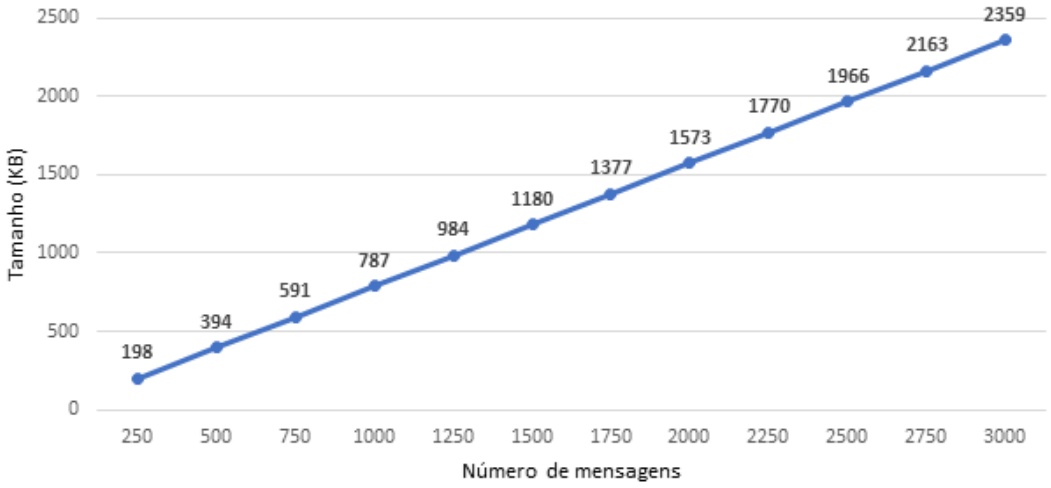
Figura 2.9: Tempo de execução das operações da biblioteca Verificatum



(a) Tamanho das provas da operação Shuffle



(b) Tamanho das provas da operação Decrypt



(c) Tamanho das provas da operação Mixing

Figura 2.10: Tamanho das provas das operações da biblioteca Verificatum

Por fim, é possível estimar o tempo de execução e verificação para quantidades diferentes de votos em dada arquitetura, além de conhecer o tamanho das provas que serão geradas. Em teoria, o tempo de execução $E(x)$ e de verificação $V(x)$ e o tamanho das provas $P(x)$ crescem linearmente com o número de votos x. Na prática, os algoritmos realizados apresentam uma pequena variação nos tempos de execução, principalmente em pontos em que é necessário gerar números aleatórios. Portanto, para obter as relações mencionadas, fez-se a regressão linear dos valores medidos, obtendo as equações aproximadas apresentadas na Tabela 2.3.

Tabela 2.3: Estimativa de desempenho e tamanho dos métodos da biblioteca Verificatum para um número x de votos. $E(x)$ corresponde ao tempo de execução, $V(x)$ ao tempo de verificação e $P(x)$ ao tamanho.

	$E(x)$ (ms)	$V(x)$ (ms)	$P(x)$ (KB)
Shuffle	2563,5 + 0,78x	1092,6 + 0,6x	0,9 + 0.78x
Decrypt	1383,6 + 0,29x	632,3 + 0,17x	0,15 + 0,32x
Mixing	2838,9 + 0,99x	1183 + 0,72x	1,1 + 0,79x

2.4.3.3. Adaptações na biblioteca

A biblioteca Verificatum foi projetada como uma implementação pura de uma mixnet. Para sua utilização no ambiente eleitoral, tornam-se necessárias algumas adaptações no código original: a exposição do método de cifração, o agrupamento de textos cifrados, e a decodificação de pontos de curvas elípticas associadas a mensagens. Esses pontos são discutidos a seguir.

Primeiramente, como o primeiro nó misturador em uma mixnet de recifração normalmente recebe as mensagens já cifradas pelos usuários, e a cifração pura não precisa ser utilizada em outros pontos do sistema, o Verificatum não expõe ao usuário o método de cifração. Além disso, o Verificatum armazena os textos cifrados de cada nó em um único arquivo. Para permitir que uma urna no cenário de auditoria cifre os votos do usuário, foram incluídos os seguintes métodos:

- A cifração de uma mensagem às claras (plaintext) com a chave pública (publicKey) produz um texto cifrado que será salvo no arquivo ciphertexts, e o número aleatório utilizado na cifração está salvo no arquivo nonce. Caso o arquivo nonce não exista, o arquivo é criado e o nonce gerado aleatoriamente é salvo nele. Cabe notar que esse mecanismo com uso de arquivo temporário não precisaria ser utilizado em uma implementação de fato, mas foi adotado no cenário de testes por ser mais próximo dos métodos já existentes na biblioteca do Verificatum.

```
vmnd -enc <publicKey> <plaintext> <ciphertexts> <nonce>
```

- O texto cifrado ciphertextin, criado com a chave pública publicKey, é acrescentado à lista de textos cifrados ciphertexts.

```
vmnd -apnd <publicKey> <ciphertexts> <ciphertextin>
```

Além disso, para fins de teste, torna-se necessário realizar a conferência dos métodos anteriores. No entanto, a cifra ElGamal construída sobre curvas elípticas codifica a mensagem a ser cifrada em um ponto da curva. Consequentemente, a decifração das mensagens retorna o ponto de curva elíptica que foi codificado. Assim, para averiguação das mensagens às claras após a decifração, também foi criado um método para decodificação das mensagens decifradas representadas por um ponto de curva elíptica:

- As mensagens decifradas messages, usado o ponto base da curva elíptica armazenado na chave publicKey, são decodificadas e salvas em decoded.

```
vmnd -dec <publicKey> <messages> <decoded>
```

2.5. Sistema E2E baseado em Criptografia Homomórfica – ElectionGuard

Esta Seção tem como objetivo apresentar a construção e a operação de um sistema de votação fim-a-fim que utiliza a biblioteca ElectionGuard, baseada em criptografia homomórfica.

O funcionamento e características do sistema serão descritos, através do uso dos blocos criptográficos previamente apresentados. Por fim, será apresentada a implementação do sistema por meio da implementação fornecida pela equipe do ElectionGuard.

2.5.1. Criptografia Homomórfica

Em 1978, Rivest, Adleman e Dertouzos propuseram uma classe de funções especiais de criptografia que eles chamaram de "homomorfismos de privacidade" [Rivest et al. 1978]. Estas funções especiais de criptografia permitem que dados cifrados possam ser operados sem a necessidade de sua decifração. A Figura 2.11 apresenta uma referência visual de criptografia homomórfica. Dados dois textos às claras m_1 e m_2 e suas cifrações c_1 e c_2, existe uma operação $\diamond$ aplicada aos textos cifrados que é equivalente a uma operação $\star$ aplicada aos textos às claras. Desta maneira, a decifração de $c_1 \diamond c_2$ é igual a $m_1 \star m_2$. Historicamente, alguns esquemas de criptografia bastante populares, como o RSA e o ElGamal, possuem uma propriedade homomórfica, como mostrado, para o último caso, na Seção 2.2.3.1.

No caso de eleições, cada eleitor tem direito a produzir um voto que representa a sua escolha e o conjunto destes votos deve ser agrupado no fim do processo, a fim de se calcular o resultado e se determinar a escolha ou escolhas vencedoras. Além disso, uma propriedade essencial de um voto é o sigilo, não devendo ser possível um terceiro determinar a escolha de um eleitor. Esta propriedade é especialmente relevante, pois ela garante ao eleitor o seu direito de livre escolha, mitigando as chances de uma eventual coerção.

Diferentemente de sistemas de votação baseados em redes de misturadores, em sistemas homomóficos os votos não são decifrados individualmente. Em outras palavras, por meio da propriedade de homomorfismo do criptosistema, um conjunto de votos cifrados são homomorficamente somados, e apenas o resultado dessa combinação de textos cifrados é então decifrado. Desta forma, um esquema criptográfico que permita a soma dos textos às claras a partir de seus textos cifrados é de especial interesse ao cenário de

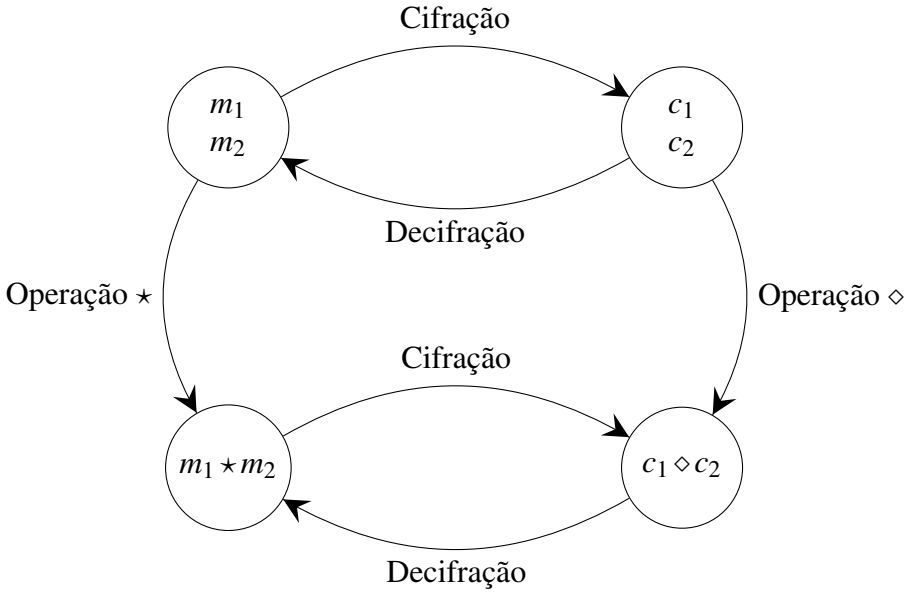


Figura 2.11: Cifração homomórfica: existe um morfismo entre a operação $\star$ realizada nos textos às claras m_1 e m_2 e a operação $\diamond$ realizada nos textos cifrados c_1 e c_2.

uma eleição. O eleitor seria capaz de criar um voto cifrado e, portanto, sigiloso, que poderia ser compartilhado publicamente. Em seguida, com todos os votos cifrados publicados, qualquer pessoa seria capaz de realizar a soma homomórfica dos votos, garantindo a utilização de todos os votos, e verificar sua igualdade com a soma cifrada publicada pela apuração oficial. Por fim, como apenas esta soma seria decifrada, o sigilo individual de cada voto seria preservado.

2.5.2. ElectionGuard

O ElectionGuard [Benaloh and Naehrig 2022] é um conjunto de ferramentas (SDK) de código aberto, cujo objetivo é o desenvolvimento de sistemas eleitorais seguros e transparentes. Para isso, o ElectionGuard utiliza um conjunto de primitivas criptográficas com a intenção de implementar a verificação fim-a-fim (E2E) ao processo. Especificamente, o ElectionGuard emprega criptografia homomórfica para garantir as propriedades fundamentais de segurança de um sistema eleitoral.

No ElectionGuard, o voto de um eleitor é constituído por um conjunto ordenado formado pela representação individual de sua escolha para cada uma das opções válidas no pleito. Caso o eleitor deseje votar em uma opção, esta opção recebe o valor 1; caso contrário, a opção recebe o valor 0. Desta maneira, em um pleito onde o eleitor pode selecionar x das n opções válidas, um voto final será representado pelo concatenamento ordenado das n opções, onde x delas apresentam o valor 1 e o restante apresentam o valor 0.

Em seguida, cada uma das opções é individualmente cifrada utilizando um esquema criptográfico homomórfico para soma, como o ElGamal Exponencial 2.2.3.2. Este voto cifrado é então adicionado a lista de votos depositados.

Por fim, uma vez finalizado o pleito, a operação homomórfica para soma é empregada em cada uma das opções individuais, para todos os votos depositados. Consequentemente, é obtido um resultado cifrado em que cada opção individual cifrada contém a soma do número de vezes que a opção foi escolhida. Portanto, a decifração única deste resultado para cada opção produz diretamente o resultado do pleito, não sendo necessária a decifração de votos individuais.

Para garantir o sigilo e integridade do processo, outros mecanismos, além do esquema criptográfico homomórfico, são necessários. Estes mecanismos serão apresentados nas seções a seguir.

2.5.2.1. Privacidade

Como apresentado anteriormente, uma opção de voto no ElectionGuard possui o valor 1 caso seja escolhida como voto ou 0 do contrário. Cada uma destas opções é individualmente cifrada e, no fim da eleição, somada homomórficamente com os outros votos para produzir o resultado cifrado final. Deste modo, somente o resultado cifrado final precisa ser decifrado para produzir o resultado da eleição. Logo, o voto individual cifrado de cada eleitor nunca é revelado, tendo sua privacidade protegida pelas garantias de segurança do criptosistema empregado.

Entretanto, uma entidade que é capaz de decifrar o resultado final também é capaz de decifrar os votos individuais, uma vez que eles estão protegidos pelo mesmo conjunto de chaves criptográficas. Portanto, esta possibilidade pode representar um risco a privacidade de um eleitor caso a entidade adote um comportamento malicioso.

A fim de mitigar este risco, o ElectionGuard faz o uso do criptosistema ElGamal com decifração limiar 2.2.3.4. Através do uso desta modalidade do esquema ElGamal, o ElectionGuard define a figura de "Guardiões da eleição". Estes Guardiões nada mais são do que entidades confiáveis, responsáveis pela geração de chaves criptográficas no criptosistema, além da decifração do resultado final. Assim, enquanto existe uma chave pública única que é utilizada para cifrar os votos durante uma eleição, a chave privada correspondente é desconhecida individualmente por cada um dos Guardiões. Como resultado, para decifrar um voto, é necessário que um número mínimo pré-estabelecido de guardiões atuem em conjunto. Portanto, para que um voto individual seja revelado, seria necessário haver um conluio entre múltiplas entidades do sistema, mitigando o risco deste ataque.

2.5.2.2. Integridade e Transparência

A fim de garantir a integridade de um processo eleitoral e ao mesmo tempo aumentar sua transparência, o ElectionGuard emprega dois mecanismos: o desafio de Benaloh e provas de conhecimento não interativas.

O desafio de Benaloh é utilizado pelo sistema para fornecer evidências ao eleitor que seu voto encontra-se corretamente representado.

Como descrito na Seção 2.3.1, no desafio de Benaloh, o eleitor pode desafiar o

sistema que cifrou o seu voto a revelar o nonce utilizado no processo. Após isso, o eleitor (ou terceiros) pode utilizar um dispositivo computacional diferente para calcular novamente o texto cifrado e o Código de Rastreio resultante. Com isto, ele é capaz de verificar se o texto cifrado resultante é igual ao produzido pelo sistema de origem.

Já as provas de conhecimento não interativas são usadas pelo sistema para fornecer evidências a todos os participantes e observadores do processo eleitoral de que as etapas envolvidas em sua operação estão sendo executadas de maneira correta. Destas provas, destacam-se as utilizadas pelo dispositivo responsável por cifrar o voto de cada eleitor e as utilizadas pelos Guardiões ao término do pleito.

Diferentemente de um sistema baseado em redes de misturadores, um sistema baseado em criptografia homomórfica nunca realiza a decifração individual de um voto. Portanto, é necessário que o dispositivo que realizou a cifração de um voto demonstre que este voto está bem formado. Isto significa provar, primeiramente, que para cada opção do voto ela foi escolhida, no máximo, uma única vez. Em seguida, prova-se que o total de opções escolhidas em um voto é igual ao total de escolhas permitidas em um voto. Como exemplo, considerando a eleição presidencial brasileira, o dispositivo precisa provar que cada candidato recebeu 0 ou 1 votos e que apenas um único candidato foi escolhido. A boa formação do voto é imprescindível para a integridade do sistema. Sem ela, seria possível ao dispositivo responsável por cifrar o voto a criação de votos com múltiplas opções escolhidas, ou múltiplas escolhas para uma única opção. Além disso, seria possível até mesmo a presença de um voto "corretor", onde números negativos são cifrados em algumas opções e múltiplos votos são cifrados em outras, produzindo um total de escolhas aparentemente correto.

Para tanto, o dispositivo que cifra o voto dos eleitores executa dois conjuntos de provas de conhecimento não interativas. O primeiro conjunto é executado para provar que cada opção de um voto recebeu o valor 0 ou 1. Para isto, o dispositivo emprega a técnica de Cramer-Darmgård-Schoenmakers 2.2.5.2 em associação com provas de Chaum-Pedersen 2.2.5.1 para valores de mensagem 0 e 1.

O segundo conjunto é executado para provar que a soma de todas as opções de um voto corresponde a um valor específico. Este valor específico é o total de escolhas permitidas em um único voto. Para isto, o dispositivo soma homomorficamente todas as opções e, utilizando novamente uma prova de Chaum-Pedersen, prova que está soma possuí o valor esperado.

Unindo os dois conjuntos de provas, qualquer participante ou observador do processo eleitoral recebe evidências de que um voto depositado é bem formado, embora não se revele o conteúdo do voto em si.

Já ao final do processo, é necessário que os guardiões provem que o resultado do pleito às claras realmente é igual ao contido no resultado final cifrado obtido pela soma homomórfica dos votos. Caso isto não fosse feito, os guadiões poderiam divulgar um resultado aleatório qualquer. Para realizar esta prova, os guardiões novamente adotam a prova de conhecimento de Chaum-Pedersen, com algumas modificações devido a decifração parcial resultante da utilização da modalidade de decifração limiar do criptosistema ElGamal.

Unindo todos as técnicas apresentadas nesta seção, o eleitor é capaz de averiguar que seu voto foi corretamente gravado. O eleitor também é capaz de observar que seu voto cifrado foi utilizado para a produção do resultado final, pois todos os votos do sistema, assim como a operação homomórfica necessária, são públicos. Logo, qualquer pessoa é capaz de produzir o resultado final utilizado pelos guardiões para a divulgação da apuração. Por fim, as provas realizadas pelo dispositivo e guardiões fornecem elementos para a garantia de que apenas votos válidos foram utilizados e de que o resultado às claras final é um produto direto do resultado cifrado final. Desta maneira, todo o processo eleitoral é verificável por qualquer observador, garantido a sua transparência.

2.5.3. Cenários para uso do ElectionGuard

Assim como no sistema baseado em redes de misturadores, propõe-se os mesmos dois cenários para o uso da ferramenta ElectionGuard, o **tradicional** e o **modelo de auditoria**.

2.5.3.1. Cenário Tradicional

Assim como no caso do uso de redes de misturadores, o cenário tradicional com o ElectionGuard também é adaptado ao voto presencial. Os componentes necessários para este cenário são uma urna eletrônica, como plataforma de votação, um sistema para a totalização dos votos, e um conjunto de entidades, que serão os guardiões do processo eleitoral.

Como preparação para a eleição, cada um dos guardiões gera seu próprio par de chaves privada e pública. As chaves públicas então são combinadas para formar a chave de cifração para a eleição. Esta chave de cifração é então previamente carregada em todas as urnas que serão utilizadas e divulgada para os eleitores e observadores.

No processo de votação, a urna funciona como plataforma de votação e é responsável por realizar corretamente a cifração dos votos dos eleitores e as provas de conhecimento descritas na Seção 2.5.2.2. Quando o eleitor confirmar suas escolhas, a urna deve emitir o Código de Rastreio, como apresentado na Seção 2.3.1.4. Caso o eleitor deseje depositar este voto, o Código de Rastreio deve ser assinado e poderá ser utilizado posteriormente como material de auditoria da integridade dos votos contabilizados.

Neste momento, como o voto não é criado por dispositivo do próprio eleitor, este pode não ter a confiança de que a urna está realizando a operação de forma correta, ou seja, ele pode desconfiar que o comprovante de votação está apresentando um valor diferente do inserido na urna. Por este motivo, o eleitor pode desafiar a urna por meio do desafio de Banaloh como descrito na Seção 2.3.1.5.

Ao final do processo de votação, inicia-se diretamente o processo de totalização. Todos os votos depositados na urna são enviados para o sistema de totalização, em conjunto com suas provas de conhecimento. Este sistema, então, verifica as provas de conhecimento de cada voto e, caso as provas sejam válidas, o soma homomorficamente ao total da eleição. Uma vez que todos os votos tenham sido somados, o resultado cifrado final é publicado e os guardiões iniciam o seu procedimento de decifração. Nesta etapa, os guardiões geram as provas de conhecimento de correta decifração do resultado e publicam o resultado às claras final em conjunto com as estas provas.

Para a auditoria, eleitores e observadores podem verificar a corretude de todas as provas de conhecimento produzidas, sejam elas referentes aos votos ou a decifração do resultado. Além disso, eleitores que desejem auditar os seus votos podem verificar que o voto associado ao seu Código de Rastreio foi utilizado para o cálculo do resultado cifrado.

2.5.3.2. Cenário de Auditoria

Assim como no caso de uso de redes de misturadores, a arquitetura do sistema neste cenário é voltada para dar poder de auditoria aos votos em uma urna eletrônica. Considera-se que esta é o suficiente para dar privacidade aos votos dos eleitores contra atacantes externos. Como resultado, o sistema é mais simplificado do que no cenário anterior, requisitando apenas dois componentes: a urna eletrônica, que executa a biblioteca ElectionGuard em sua totalidade, e um sistema de totalização para votos às claras.

O processo de preparação para eleição ocorre individualmente para cada urna. Cada uma das urnas é considerada o único guardião de uma eleição que abrange apenas os eleitores desta urna. A urna cria um par de chaves pública-privada, onde a chave pública será utilizada como a chave de cifração para a urna. Adicionalmente, a chave de cifração é divulgada para todos os interessados.

O processo de votação ocorre de maneira análoga ao cenário tradicional, os eleitores tem seus votos cifrados pela urna, que também gera as provas de conhecimento para eles. Os votos podem então serem desafiados ou depositados, neste último caso sendo realizadas as assinaturas dos Códigos de Rastreio.

A diferença principal dos cenários ocorre ao final do processo de votação. Nesta etapa, a urna soma homomorficamente todos os votos e realiza a decifração do resultado final, produzindo a apuração para aquela urna. A prova de correta decifração é igualmente produzida e todo o material é enviado ao sistema de totalização.

O sistema de totalização, por sua vez, apenas verifica as provas de conhecimento produzidas pela urna e, em caso de elas estarem corretas, soma o resultado já decifrado da urna ao total da eleição. Este processo é bastante similar ao já existente no sistema de votação brasileiro, onde o Tribunal Superior Eleitoral (TSE) soma os resultados parciais de cada urna no país, os Boletins de Urna (BU), para totalizar a eleição. Desta maneira, assim como no caso de uso de redes de misturadores, o cenário de auditoria aparenta ser uma escolha natural para o contexto de votação brasileiro.

Também de maneira análoga ao caso de uso de redes de misturadores e através da mesma argumentação, neste cenário a confidencialidade dos votos obtida é essencialmente a mesma do que em uma urna eletrônica tradicional brasileira.

2.5.3.3. Instalação

O núcleo da biblioteca possui implementações nas linguagens Python e C++. No entanto, somente a versão Python possui uma API e uma interface de usuário. Para fins de instalação do SDK e seus adicionais, são referenciados os seguintes endereços:

A versão do SDK em linguagem Python está disponível em `https://github.com/microsoft/electionguard-python` e sua versão C++ em `https://github.com/microsoft/electionguard-cpp/`. A API Python pode ser encontrada em `https://github.com/microsoft/electionguard-api-python`. Uma implementação de rederência da interface de usuário está disponível em `https://github.com/microsoft/electionguard-ui/tree/main`.

2.5.3.4. Avaliação de eficiência

Diferentemente de um sistema de votação baseado em redes de misturadores, um sistema baseado em criptografia homomórfica não necessita de um procedimento especial após o período de votação para garantir privacidade ou integridade. Estas garantias são conferidas no momento de cifração do voto e criação das provas de conhecimento associadas a ele. Deste modo, para avaliar a eficiência da biblioteca ElectionGuard, optou-se por aferir o tempo necessário para a etapa de criação do voto. Esta etapa é a que impacta fundamentalmente no tempo de resposta do sistema para o eleitor, já que este deve esperar sua conclusão para receber seu Código de Rastreio e realizar a escolha entre depósito ou desafio.

O hardware utilizado no teste é o mesmo apresentado na Seção 2.4.3.2:

- Processador: Intel Atom Elkhart Lake Processor, J6412/X6413E
- Memória: 2 x 8GB SODIMM DDR4 3200Mhz
- Disco: SSD Crucial P2 500 GB 3D NAND NVMe PCIe M.2 (Sequential Read - 2300 MB/s, Sequential Write - 940 MB/s)
- Sistema operacional: Ubuntu 22.04.1 LTS

Foi utilizada a versão 1.3.0 de 12/10/2021 da biblioteca ElectionGuard em Python, sendo executada com Python versão 3.9.2.

Para os votos em si, foram considerado votos com 100, 200, 500, 1000, 1500, 2000, 2500 e 3000 candidatos. É importante observar que o tempo de cifração e criação de provas é linearmente dependente ao número de candidatos. Isto ocorre pois cada candidato acrescenta é cifrado individualmente e requer provas igualmente individuais, com exceção da prova de total de escolhas permitidas.

Também é importante notar que o número de candidatos considerados é extremamente alto em alguns cenários. Esta escolha foi feita para considerar a eleição para vereadores na cidade de São Paulo, que apresenta esta quantidade de candidatos. Apesar de não se esperar que os tempos para estes cenários fossem factíveis para uma implementação de mundo real, eles permitem com que se faça uma interpolação dos resultados para obter o tempo esperado por candidato na disputa. Consequentemente, inicialmente os resultados destes grandes cenários são apresentados de maneira integral na Figura 2.12 Posteriormente, são apresentados os tempos interpolados para cada candidato em uma cédula de votação, na Tabela 2.4.

A fim de obter um tempo de resposta menor do que 2 segundos, o número máximo de candidatos possíveis é igual a 11. Esta quantidade de candidatos é adequada para eleições de cargo majoritário, como presidente e governador. Entretanto, para um cenário de cargos minoritários, especialmente no Brasil, esta quantidade de candidatos é insuficiente.

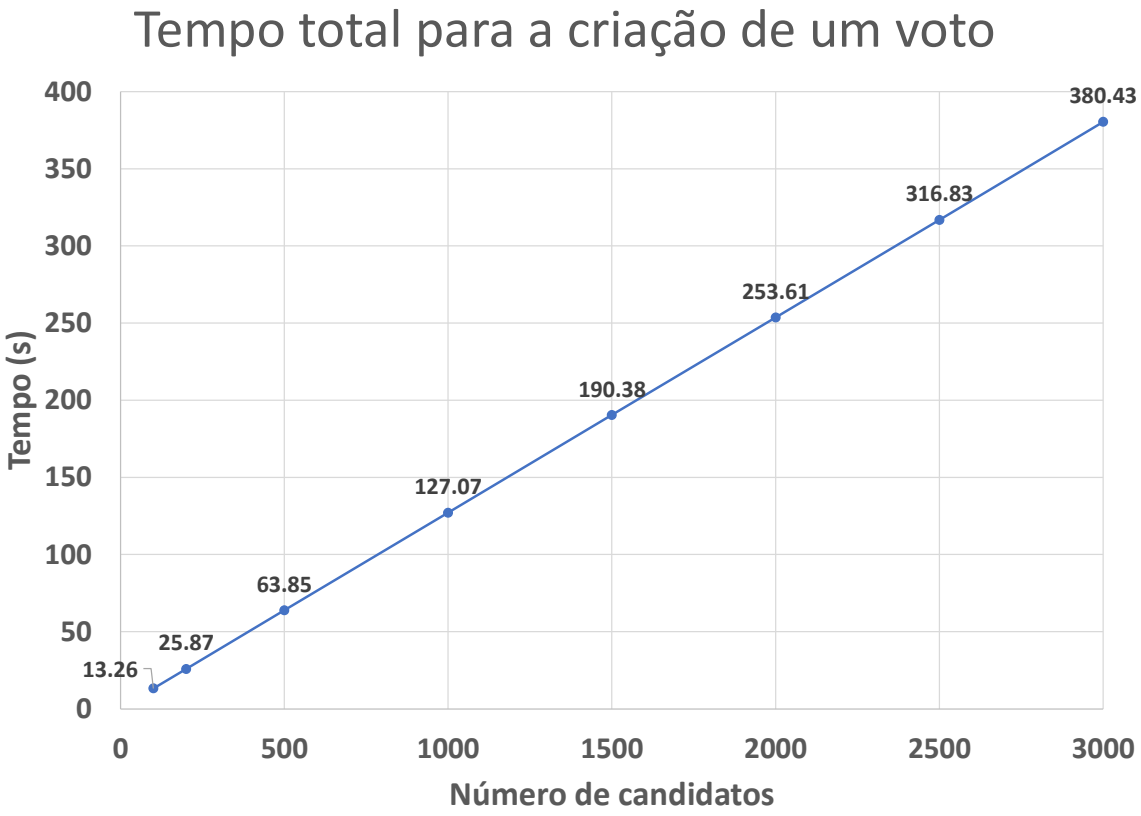


Figura 2.12: Tempo para a criação de um voto na biblioteca ElectionGuard para cédulas com múltiplos candidatos. O tempo inclui a cifração e a elaboração das provas de conhecimento.

De maneira análoga, a Figura 2.13 apresenta o tamanho de um voto gerado pela biblioteca ElectionGuard para os mesmos cenários anteriores. O tamanho dos votos para cada candidato também é interpolado e incluído na Tabela 2.4

Para o cenário de 11 candidatos abordado anteriormente, o voto de cada eleitor possui o tamanho de aproximadamente $0,219$ MB. Desta maneira, para cada 1000 votos, o sistema ElectionGuard produz cerca de 219 MB de dados.

Tabela 2.4: Interpolação do tempo necessário para a criação e do tamanho final de um voto no ElectionGuard para cada candidato.

	Tempo (s)	Tamanho (MB)
Inicial	$0,545$	$0,13427$
Por candidato	$0,127$	$0,00768$

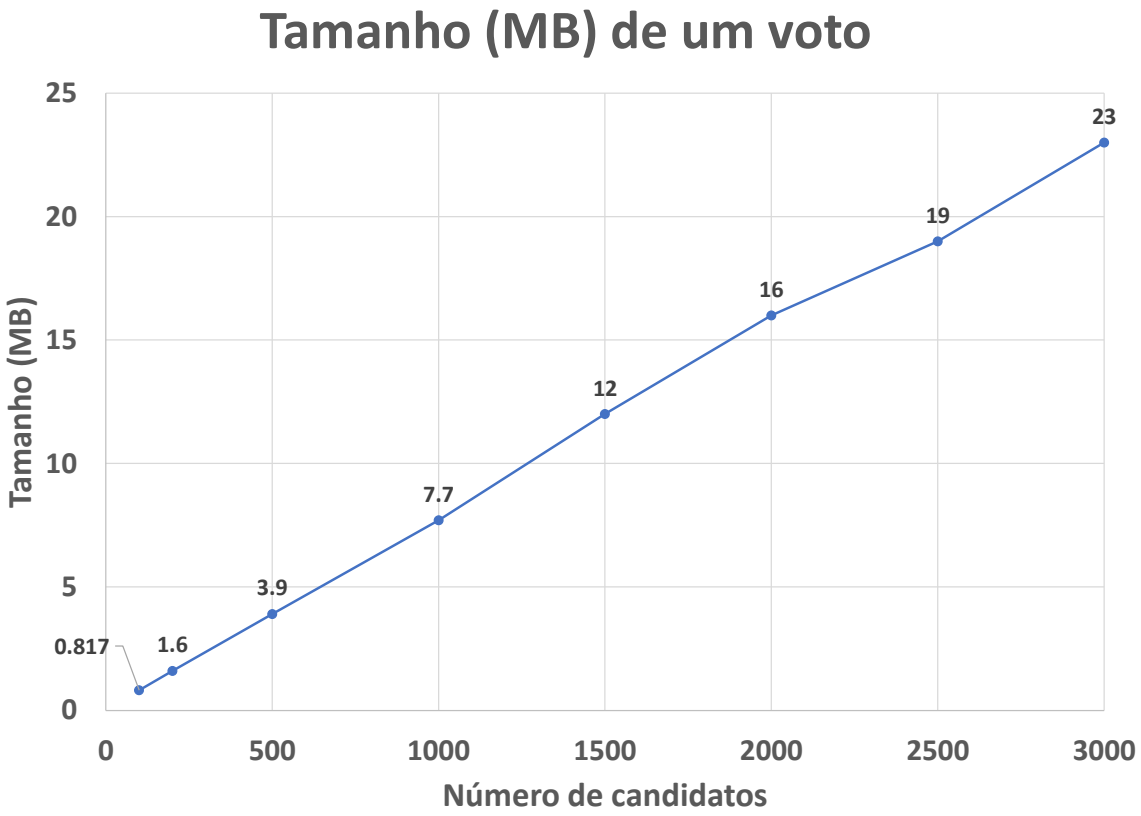


Figura 2.13: Tamanho de um voto na biblioteca ElectionGuard para cédulas com múltiplos candidatos. O tamanho inclui a cifração e as provas de conhecimento.

2.6. Comparação entre Sistemas baseados em Redes de Misturadores e Homomorfismo

Nesta Seção, é realizada uma comparação entre os dois modelos de votação fim-a-fim. Inicialmente, a necessidade de tratar os dados após o término da votação é comparada, pois esta é a principal diferença entre os dois métodos. Em seguida, são feitas comparações relativas a privacidade, integridade e desempenho obtidas pelos esquemas estudados.

2.6.1. Tratamento de dados após encerramento da votação

A principal diferença entre um esquema que utiliza redes de misturadores e criptografia homomórfica é a necessidade de tratar os dados após o encerramento da votação.

Em um sistema de redes de misturadores, quando a eleição se encerra, é necessário que os votos cifrados sejam transmitidos a um ou mais nós misturadores antes de serem entregues ao sistema de totalização. Este é um passo fundamental, pois como os votos serão individualmente decifrados para a apuração, é preciso quebrar o vínculo entre os votos inicialmente cifrados e os eleitores que o adicionaram ao sistema. Desta maneira, existe um tráfego de dados extra, além de um pós processamento relativo ao processo de mixagem dos votos pelos nós misturadores.

Já em um sistema baseado em criptografia homomórfica, os votos cifrados vinculados a eleitores nunca são individualmente decifrados. Logo, após o encerramento da

votação, estes dados podem ser transmitidos diretamente ao sistema de totalização.

Consequentemente, em um sistema que utiliza redes de misturadores, existe um tempo adicional entre o fim da votação e o início da apuração. A superfície de ataque do sistema de redes de misturadores também é maior, visto que os nós misturadores são entidades independentes e podem ser individualmente atacadas. Entretanto, o tempo de processamento extra, relativo a aplicação do método "*mixing*" aos votos cifrados, não é grande, como indicado na Seção 2.4.3.2. Da mesma maneira, como é possível existirem mais nós misturadores do que a quantidade necessária, um eventual ataque em um ou alguns nós misturadores pode ser mitigada pelo uso dos demais nós.

2.6.2. Comparação de privacidade

A privacidade do eleitor em um sistema que usa redes de misturadores é dependente de ao menos um nó misturador honesto e da confiança na entidade, ou conjunto de entidades, que realiza a decifração dos votos.

No primeiro caso, ao menos um nó misturador honesto é necessário a fim de quebrar o vínculo entre os votos cifrados antes do processamento pelos nós e os votos cifrados após a execução do processo de mixagem pelos nós. Caso haja um conluio entre todos os nós misturadores, eles são capazes de reverter o processo de mistura e, assim que os votos na saída forem decifrados pelo sistema de totalização, associar um voto às claras a um eleitor.

Além disso, é preciso confiar na entidade, ou conjunto de entidades, que realiza a decifração dos votos misturados. Como a chave de decifração é a mesma tanto para os votos na entrada da rede de misturadores quanto para os votos na saída, a entidade responsável pela decifração poderia decifrar, adicionalmente, os votos não misturados. Consequentemente, a entidade seria capaz de associar votos às claras com eleitores. Este ataque pode ser mitigado pelo uso do sistema ElGamal com decifração limiar. Deste modo, seria necessário um conluio das entidades detentoras das chaves de decifração para executar este ataque.

No caso de um sistema baseado em criptografia homomórfica, a privacidade depende da confiança nos Guardiões do sistema, ou seja, nas entidades que realizam a decifração dos votos. A razão da necessidade de confiança nestas entidades é exatamente a mesma do modelo baseado em redes de misturadores: as entidades que possuem as chaves de decifração podem decifrar os votos individuais de cada eleitor, em adição a soma homomórfica. Igualmente ao explicado anteriormente, este ataque é mitigado pelo uso do sistema ElGamal com decifração limiar, ou seja, pelo uso de múltiplos Guardiões.

Como resultado da comparação de privacidade entre os dois sistemas de votação estudados, é exigido a confiança extra no sistema com uso de redes de misturadores de que exista ao menos um nó misturador honesto. Contudo, a presença de diversos nós misturadores e a possível inclusão e troca destes nós a qualquer momento do processo eleitoral reduz um possível risco de privacidade existente no modelo.

2.6.3. Comparação de integridade

Existe uma equivalência em relação a integridade entre os dois métodos estudados neste minicurso.

No sistema baseado em redes de misturadores, a integridade depende das provas de conhecimento zero de mistura, recifração e decifração, além do uso do desafio de Benaloh e da construção do Código de Rastreio.

Já no sistema que utiliza criptografia homomórfica, a integridade depende das provas de conhecimento zero de que os votos são bem formados e das provas de correta decifração, além de, de maneira igual ao anterior, do uso do desafio de Benaloh e da construção do Código de Rastreio.

Como a segurança e confiabilidade das provas utilizadas em ambos os métodos são similares, não existe uma diferença significativa em relação a integridade do processo em razão destas primitivas. Além disso, como o desafio de Benaloh e a construção do Código de Rastreio ocorre de maneira igual nos dois sistemas, também não existe uma diferença em relação a integridade neste ponto. Consequentemente, a integridade nos dois modelos estudados para a implementação de um sistema fim-a-fim não possuí uma diferença significativa notável.

2.6.4. Comparação de desempenho

A comparação de desempenho entre os dois modelos de sistema deve levar em conta o tempo gasto e o volume de dados gerados por cada um deles.

Em relação ao tempo gasto, o sistema baseado em redes de misturadores concentra este tempo no período posterior ao encerramento da votação. Para cada **voto** lançado por um eleitor, um nó misturador leva cerca de 1ms a mais de tempo para realizar a mistura.

Já em um esquema que usa criptografia homomórfica, o tempo gasto concentra-se no período de lançamento do voto, para cada eleitor individual. Para cada **candidato** extra existente em uma cédula de votação, o sistema necessita de 127ms a mais para criar o voto.

Portanto, caso o tempo seja o parâmetro crítico na escolha entre os dois métodos de implementação de um esquema de votação fim-a-fim, uma eleição com poucos candidatos é ideal para um sistema baseado em criptografia homomórfica, enquanto que uma eleição com poucos eleitores é recomendado para o modelo de redes de misturadores. Entretanto, o número de eleitores causa um impacto menor em uma rede de misturadores do que o número de candidatos causa em um sistema com uso de criptografia homomórfica. Este fato é ainda mais considerável pois o impacto do tempo em um esquema homomórfico recai no eleitor, que necessita esperar com que o sistema encerre a cifração de seu voto. Logo, para cada cenário de eleição é necessário um estudo levando em consideração os parâmetros de candidatos existentes e de número de eleitores.

No caso da eleição brasileira, que conta com muitos candidatos e muitos eleitores, o uso de um sistema baseado em criptografia homomórfica é inviável, pois o tempo de resposta do sistema para o eleitor, durante a criação do voto, é extremamente alto, chegando a ordem de minutos no caso de uma eleição para vereador.

Em relação ao volume de dados gerados, um sistema baseado em redes de misturadores cria uma prova de conhecimento no processo de mistura, que aumenta em aproximadamente $0,79$ KB para cada **voto** extra no sistema. Como cada nó misturador produz uma prova independente, o resultado acima é multiplicado pelo número de **nó misturadores** utilizados. Por outro lado, um esquema que utiliza criptografia homomórfica cria uma prova de conhecimento no processo de votação, que aumenta em aproximadamente $7,7$ KB para cada **candidato** extra na cédula. Como cada eleitor produz um voto, o resultado acima também é multiplicado pelo número de eleitores, ou **votos**, no sistema.

Consequentemente, um sistema baseado em criptografia homomórfica gera um volume de dados maior do que um esquema que utiliza rede de misturadores, ao menos que o número de nós misturadores seja bastante elevado. No caso de uma eleição com dois candidatos, um nó misturador cria uma prova de tamanho 790 MB para 1 milhão de eleitores. Enquanto isso, um sistema homomórfico produz um voto de tamanho $0,14963$ MB para cada eleitor, totalizando 149.630 MB para 1 milhão de eleitores. Desta maneira, seriam necessários cerca de 190 nós misturadores para que o volume de dados fosse equivalente.

É importante observar que no caso de uso de um sistema de rede de misturadores, os votos cifrados, e apenas os votos cifrados, devem ser repassados entre os diversos nós misturadores, gerando um tráfego extra de rede, Entretanto, o tamanho de um voto cifrado, sem provas de conhecimento, é pequeno. Logo, mesmo levando em conta um potencial uso extra de rede no cálculo acima apresentado, um sistema baseado em redes de misturadores continua necessitando de uma banda de comunicação consideravelmente inferior a de um esquema de criptografia homomórfica.

2.7. Considerações Finais

Neste minicurso foi realizada uma apresentação dos principais conceitos de um sistema de votação fim-a-fim, ou E2E. Primeiramente, foram apresentadas as principais primitivas criptográficas utilizadas na construção dos sistemas fim-a-fim de interesse para este documento.

Em seguida, foi realizada uma apresentação geral sobre sistemas E2E, com os principais mecanismos utilizados e as principais características oferecidas por este método.

Uma vez realizada a apresentação geral da teoria de sistemas fim-a-fim, foram discutidas as duas implementações mais populares desta técnica: através do uso de redes de misturadores e através do uso de criptografia homomórfica.

Por fim, foi feita uma breve comparação entre os dois métodos de implementação, destacando suas diferenças e semelhanças em relação as principais propriedades de segurança desejadas em um sistema de votação. Uma pequena comparação em relação ao desempenho dos dois sistemas também foi realizada.

Um dos principais desafios em relação a sistemas fim-a-fim é a explicação de seus conceitos para uma população leiga. A população, para confiar no sistema de votação, precisa acreditar que os mecanismos de integridade e privacidade implementados pelas primitivas criptográficas realmente funcionam. Como estes mecanismos envolvem o uso

de matemática complexa, mecanismos de explicação mais simples, como alegorias, devem ser desenvolvidos.

Um outro desafio é o estudo do cenário de votação ao qual o sistema E2E será aplicado a fim de determinar qual implementação deve ser usada. Embora existam casos nos quais a adoção de um ou outro sistema é evidente, também existem cenários nos quais ambas as implementações podem ser consideradas. Consequentemente, cada cenário necessita ser individualmente analisado a fim de se adotar a melhor escolha possível.

Referências

[Adida 2008] Adida, B. (2008). Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348.

[Benaloh 2006] Benaloh, J. (2006). Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5.

[Benaloh and Naehrig 2022] Benaloh, J. and Naehrig, M. (2022). Electionguard specification 1.1. Technical report, Microsoft Research. `https://www.electionguard.vote/spec/`.

[Berlinski et al. 2021] Berlinski, N., Doyle, M., Guess, A. M., Levy, G., Lyons, B., Montgomery, J. M., Nyhan, B., and Reifler, J. (2021). The effects of unsubstantiated claims of voter fraud on confidence in elections. *Journal of Experimental Political Science*, pages 1—16.

[Chaum et al. 2008] Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R. L., Ryan, P. Y., Shen, E., Sherman, A. T., et al. (2008). Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. *EVT*, 8(1):13.

[Chaum et al. 2021] Chaum, D., Carback, R. T., Clark, J., Liu, C., Nejadgholi, M., Preneel, B., Sherman, A. T., Yaksetig, M., Zhang, B., et al. (2021). Votexx: Coercion resistance for the real world (preliminary extended abstract). *UMBC Student Collection*.

[Chaum and Pedersen 1992] Chaum, D. and Pedersen, T. P. (1992). Wallet databases with observers. In *Annual international cryptology conference*, pages 89–105. Springer.

[Chaum et al. 2005] Chaum, D., Ryan, P. Y., and Schneider, S. (2005). A practical voter-verifiable election scheme. In *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*, pages 118–139. Springer.

[Chaum 1981] Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90.

[Commission 2023] Commission, U. E. A. (2023). End to End (E2E) protocol evaluation process. https://www.eac.gov/voting-equipment/end-end-e2e-protocol-evaluation-process. Acessado em 2 de abril de 2023.

[Cramer et al. 1994] Cramer, R., Damgård, I., and Schoenmakers, B. (1994). Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer.

[ElGamal 1985] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

[Fiat and Shamir 1986] Fiat, A. and Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer.

[Haenni et al. 2017] Haenni, R., Locher, P., Koenig, R., and Dubuis, E. (2017). Pseudo-code algorithms for verifiable re-encryption mix-nets. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 370–384. Springer.

[Hubbers et al. 2005] Hubbers, E., Jacobs, B., and Pieters, W. (2005). Ries-internet voting in action. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, volume 1, pages 417–424. IEEE.

[Jakobsson et al. 2002] Jakobsson, M., Juels, A., and Rivest, R. L. (2002). Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium (USENIX Security 02)*.

[Kelsey et al. 2010] Kelsey, J., Regenscheid, A., Moran, T., and Chaum, D. (2010). Attacking paper-based e2e voting systems. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, pages 370–387. Springer.

[Lindeman and Stark 2012] Lindeman, M. and Stark, P. B. (2012). A gentle introduction to risk-limiting audits. *IEEE Security & Privacy*, 10(5):42–49.

[Lindeman and Stark 2020] Lindeman, M. and Stark, P. B. (2020). Tools for comparison risk-limiting election audits. https://www.stat.berkeley.edu/~stark/Vote/auditTools.htm. Acessado em 19 de julho de 2023.

[Nicas et al. 2022] Nicas, J., Milhorance, F., and Ionova, A. (2022). How Bolsonaro built the myth of stolen elections in Brazil. https://www.nytimes.com/interactive/2022/10/25/world/americas/brazil-bolsonaro-misinformation.html.

[Park et al. 1994] Park, C., Itoh, K., and Kurosawa, K. (1994). Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic*

Techniques Lofthus, Norway, May 23–27, 1993 Proceedings 12, pages 248–259. Springer.

[Perez 2021] Perez, E. (2021). Hacking to save democracy. *Voting Village - DEF CON*.

[Popoveniuc and Hosp 2010] Popoveniuc, S. and Hosp, B. (2010). An introduction to punchscan. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, pages 242–259. Springer.

[Rivest et al. 1978] Rivest, R., Adleman, L., and Dertouzos, M. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

[Rivest 2008] Rivest, R. L. (2008). On the notion of 'software independence'in voting systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3759–3767.

[Ryan et al. 2005] Ryan, P., Peacock, T., et al. (2005). Prêt à voter: a system perspective. *School of Computing Science Technical Report Series*.

[Sampigethaya and Poovendran 2006] Sampigethaya, K. and Poovendran, R. (2006). A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181.

[Terelius and Wikström 2010] Terelius, B. and Wikström, D. (2010). Proofs of restricted shuffles. In *Progress in Cryptology–AFRICACRYPT 2010: Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings 3*, pages 100–113. Springer.

[Zagórski et al. 2013] Zagórski, F., Carback, R. T., Chaum, D., Clark, J., Essex, A., and Vora, P. L. (2013). Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *International Conference on Applied Cryptography and Network Security*, pages 441–457. Springer.