Esquema de votação seguro e transparente através de encriptação homomórfica

Pedro Vinícius Macêdo de Araújo[1], **Antônio de Abreu Batista Jr**[1], **Mario A. Gazziro**[2]

[1]Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses, 1966 - Vila Bacanga, São Luís - MA, 65080-805, Brasil

[2]Universidade Federal do ABC (UFABC)
Av. dos Estados, 5001 - Bangú, Santo André - SP, 09210-580, Brasil

pedro.vma96@gmail.com, antonio.batista@ufma.br, mario.gazziro@ufabc.edu.br

Abstract. *Democracy depends on a large portion of voter's trust in the integrity of the process of electing their representatives. Thus, voting schemes need to be transparent and still guarantee the secrecy of the vote to ensure the integrity of elections. In this article, we propose a voting scheme using homomorphic encryption and commitment schemes. The system holds not only the confidentiality of the voters' vote but also to allow them to audit the outcome of the election. The experimental results show that the system performs well for elections with small numbers of voters.*

Resumo. *A democracia depende em uma parcela grande da confiança dos eleitores na integridade do processo de escolha de seus representantes. Consequentemente, esquemas de votação precisam ser transparentes e ainda garantir o sigilo do voto a fim de garantir a integridade de uma eleição. Neste artigo, propomos um esquema de votação usando encriptação homomórfica e esquemas de comprometimento. O sistema garante não só a privacidade do voto dos eleitores, mas também permite que eles auditem o resultado da eleição. Os resultados experimentais mostram que o sistema tem bom desempenho para eleições com pequeno número de eleitores.*

1. Introdução

A democracia depende em uma parcela grande da confiança dos eleitores na integridade das eleições. No entanto, a integridade de uma eleição pode ser afetada de muitas formas [National Academies of Sciences and Medicine 2018], e.g., contabilizando votos ilegítimos ou não permitindo auditar o resultado da eleição. A fim de lidar com essas e outras ameaças, idealmente, um sistema de votação deve ser tanto seguro quanto transparente. O sistema, entre outras garantias, deve permitir que qualquer eleitor possa mostrar que o total de votos de um candidato foi ou não alterado e que somente votos de eleitores legítimos, incluindo o seu, são considerados na contagem.

Recentemente, um número de trabalhos tem sido proposto a fim de garantir não só o sigilo do voto, mas também a transparência do processo de votação. [Lima et al. 2017] desenvolveram um esquema de votação em que a contagem de votos é feita publicamente, lendo voto a voto, através da leitura de códigos QR. Por outro lado, [Azougaghe et al. 2015, Peng and Bao 2011] criaram sistemas de votação usando técnicas de encriptação homomórfica [Azougaghe et al. 2015, Peng and Bao 2011], em que a

combinação de um grupo de votos encriptados revela o resultado da eleição, sendo que nenhuma cédula individual tem seu conteúdo revelado.

Também, usando tais técnicas, [Hirt and Sako 2000] desenvolveram um esquema de votação em que eleitores honestos e desonestos não podem provar como eles votaram. Por sua vez, [Yang et al. 2018] criaram um esquema de votação pela internet inspirado no chamado voto por aprovação [Brams and Fishburn 2005]. Recentemente, [Graaf 2017] propôs um esquema de votação que combina técnicas de encriptação homomórfica e esquemas de comprometimento[1] em que a confidencialidade do voto é garantida a longo prazo.

Finalmente, é importante notar que apesar do número elevado de abordagens propostas na literatura, sistemas de votação combinando encriptação homomórfica e esquemas de comprometimento são pouco explorados. Neste artigo, propomos um sistema de votação que combina encriptação homomórfica, esquemas de comprometimento e provas de conhecimento parcial[2]. O sistema garante não só o sigilo do voto, mas também permite que os eleitores auditem o resultado da eleição. Além disso, ele permite a coexistência da votação eletrônica e física (em papel), o que aumenta a transparência do processo de votação.

O restante deste artigo está estruturado da seguinte forma: a Seção 2 apresenta o nosso modelo de segurança; a Seção 3 apresenta o embasamento teórico do nosso esquema de votação; a Seção 4 apresenta o esquema de votação proposto; a Seção 5 apresenta uma análise empírica do desempenho dos algoritmos de verificação e totalização dos votos, além das provas matemáticas da segurança computacional do esquema; finalmente, a Seção 6 apresenta as conclusões deste trabalho e possíveis direções para futuras pesquisas.

2. Modelo de segurança

Neste trabalho, consideramos que um esquema de votação eletrônica é seguro e transparente se ele satisfaz as seguintes propriedades:

1. Somente eleitores válidos podem votar;
2. Cada eleitor só pode votar uma vez;
3. Sigilo total do voto, impossibilitando a dedução de qualquer informação sobre a escolha do eleitor, mesmo com a conivência deste;
4. O eleitor pode, após o voto, verificar se ele é realmente válido;
5. O eleitor pode se convencer que seu voto realmente foi apurado;
6. Ninguém deve conseguir alterar ou remover os votos na urna ou incluir votos ilegítimos;
7. Todos os votos devem permanecer secretos até o fim da votação;
8. Todos os votos na urna, e somente esses, serão incluídos na contagem;
9. A contagem dos votos deve ser pública;
10. Deve ser possível auditar a contagem.

[1]Um esquema de comprometimento é uma primitiva criptográfica que permite que uma parte se comprometa com um valor escolhido enquanto mantém esse valor oculto de outras partes, podendo revelar o valor comprometido mais tarde.

[2]O código-fonte dos algoritmos do sistema de votação pode ser acessado no endereço: `https://github.com/pdrvncs/UrnaPrototipo`

O funcionamento do esquema proposto depende das suposições:

1. Um oponente possui poder computacional limitado e só pode corromper uma parte das autoridades;
2. Um oponente pode entrar em contato com o eleitor ou monitorá-lo durante o processo de votação. Porém, supomos que o adversário não pode monitorar ou interagir com o eleitor continuamente durante a sua permanência nas cabines de votação;
3. O *software* da urna é assinado digitalmente pelas autoridades;
4. As máquinas são projetadas para executarem unicamente o *software* da urna com a assinatura digital das autoridades.

3. Conceitos básicos

ElGamal

O ElGamal é um cripto-sistema de chave pública com sua segurança proveniente do problema do logaritmo discreto. O esquema pode ser definido sobre qualquer grupo cíclico com ordem p e um gerador g. O ElGamal possui a propriedade de que uma mesma mensagem m produz diferentes cifras desde que o valor aleatório s usado no processo de cifração seja diferente, possibilitando assim, no nosso trabalho, que eleitores com votos idênticos apresentem cifras diferentes para seus respectivos votos.

Encriptação Homomórfica

A fim de permitir a contagem dos votos sem ter acesso ao seus conteúdos, faz-se necessário o uso de um cripto-sistema que apresente propriedade homomórfica. Um cripto-sistema de cifração $E()$ é dito homomórfico se para dados $E(x)$ e $E(y)$, pode-se obter $E(x \oplus y)$ sem que tenhamos que conhecer x ou y, para alguma operação $\oplus$. Com uma modificação no método de cifração do ElGamal, é possível obter uma propriedade de homomorfismo aditivo, tornando possível computar $E(x) \times E(y) = E(x+y)$. Esse método é conhecido como ElGamal Exponencial.

Compartilhamento de segredo

Nesta proposta, divide-se o segredo x de acesso ao sistema de votação entre as autoridades a fim de dificultar ataques coercitivos e/ou de suborno. A ideia é que o segredo seja dividido em n partes e para recuperá-lo seja necessária a presença de pelo menos um número $t < n$ de partes. A seguir descrevemos o procedimento adotado para obter x:

1. Escolha um primo p grande e um elemento gerador g do grupo cíclico ${\mathbb{Z}_p}^*$.
2. Escolha aleatoriamente k partes $y_1, y_2, \cdots, y_k$ a partir de $\{\mathbb{Z}^*_{p-1} - 1\}$.
3. Calcule $x = \sum_{i=1}^{k} y_i \bmod p$ e defina $\beta \equiv g^x \bmod p$.
4. Defina (p, g, β) com a chave pública e x como a chave privada.

Esquemas de comprometimento

São primitivas criptográficas que permitem que um emissor possa se comprometer com um valor escolhido, ao mesmo tempo que mantém esse valor escondido do receptor, podendo revelar esse valor depois sem permitir que o emissor mude de ideia e revele um valor diferente do comprometido. Neste esquema de votação, a primitiva é usada

para manter a privacidade do voto incondicionalmente. Dada um voto m e um valor $s \in \{\mathbb{Z}^*_{p-1} - 1\}$ escolhido aleatoriamente, o ***bit-commitment*** do voto ***m*** é dado pela equação $BitCommitment(m, s, \overbrace{(g, \beta, p)}^{publickey}) = g^s \beta^m \bmod p$, de modo que se s for descartado, então será preciso resolver o problema do logaritmo discreto para recuperar o voto m.

Prova de Conhecimento Parcial

Prova de conhecimento parcial é um método que garante que uma parte consiga provar para outra parte alguma afirmação sem revelar nenhuma outra informação além disso. Assim, a fim de evitar que votos com valores inválidos sejam aceitos na contagem, faz-se necessário uma forma de provar que os votos encriptados equivalem a votos válidos, mas sem comprometer o sigilo dos votos encriptados. $PPK(c, publicKey, s, m_1, m_2)$ é a função que gera uma prova de conhecimento parcial para um valor cifrado c, utilizando $publicKey$ e o valor secreto s. A prova mostra que c decifrado equivale a m_1 ou m_2. Já $PPK_{Test}(z, publicKey)$ verifica se uma prova de conhecimento parcial z é válida, retornando ***TRUE*** se os testes passarem ou ***FALSE***, caso contrário. Para mais detalhes sobre o método de prova de conhecimento parcial usado neste trabalho consultar [Yang et al. 2018].

4. Esquema de Votação Eletrônica Proposto

Levando em consideração o modelo de segurança e os blocos de construção introduzidos, apresentamos o esquema de votação proposto. O esquema é composto de quatro fases e envolve três participantes: eleitores, autoridades e o ***Counter***[3].

4.1. Fase de Configuração da Eleição

Essa fase engloba dois procedimentos principais, o primeiro deles é a distribuição das partes (pedaços) da chave privada (segredo) usada pelas autoridades para publicação oficial do resultado final da eleição. O segundo é a configuração para o estado inicial das estruturas de dados (representando a tabela a ser publicada) que são mantidas pelo ***Counter***. A Tabela 1 mostra quais são estas estruturas.

Tabela 1. Estruturas de dados mantidas pelo $Counter$. Elas são atualizadas pelo algoritmo 3. Essas estruturas são mantidas encriptadas usando ElGamal com uma chave compartilhada.

Estrutura de dados	Descrição
$R_j \quad j = \{1, \cdots, N\}$	contém a soma de votos para cada candidato j.
$T_j \quad j = \{1, \cdots, N\}$	contém a soma $\sum_{i=1}^{n} s_{ij}$ para cada candidato j.

4.2. Fase de Registro do Eleitor

O eleitor é identificado usando biometria. Se sua elegibilidade for confirmada, ele deve assinar a lista de presença, com o mesário como testemunha. Então, o eleitor recebe permissão para votar.

[3]Um servidor externo que conta votos válidos.

4.3. Fase de Votação

Nesta fase, o eleitor escolhe seu candidato na máquina de votação que gera o voto físico e digital que é processado pelo *Counter*. No esquema proposto, o voto v_i do eleitor i em seu candidato j é representado por uma linha da matriz $A_{I \times N} = (a_{ij})$, i.e., $v_i = \{a_{i1}, a_{i2}, \cdots a_{iN}\}$ em que I é o número de eleitores e N o número de candidatos, e cada elemento da matriz é definido como segue:

$$a_{ij} = \begin{cases} 2 & se\ i\ votou\ em\ j \\ 1 & \text{caso contrário} \end{cases} \tag{1}$$

Nessa representação, um voto de um eleitor i é válido se:

$$\sum_{j=1}^{N} a_{ij} = \begin{cases} N+1 & \text{se } i \text{ escolheu um candidato} \\ N & \text{se } i \text{ votou nulo ou branco} \end{cases} \tag{2}$$

Quando um eleitor confirma a sua escolha de candidato, a máquina, já previamente configurada com a chave pública $publicKey$, executa o Algoritmo 1 gerando a tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$. A Tabela 2 define cada um desses elementos da tupla. Ao final, a máquina imprime uma cédula em papel contendo três segmentos como mostrado na Figura 1. A cédula é separável nas duas regiões tracejadas. O segmento 1 contém o voto em texto em claro, o segmento 2 contém um recibo contendo a tupla $< ID_i, H_i, B_i >$ e o segmento 3 contém a tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$ ou s_i.

Tabela 2. Valores enviados para o $Counter$ pela urna física para o registro do voto de um eleitor.

Notação	Descrição
ID_i	Identificação do eleitor.
c_i	A escolha do eleitor cifrada através do ElGamal Exponencial.
S_i	s_i cifrado através do ElGamal tradicional.
B_i	BitCommitment para o voto v_i.
X_i	Prova de que $\text{c}_{ij} \in \text{c}_i$ é 1 ou 2 para $j = \{1, \cdots, N\}$.
Y_i	Prova de que c_i é N ou $N+1$.
H_i	Hash(B_i).

Após a confirmação do voto pelo eleitor, a máquina pergunta ao eleitor se ele deseja verificar a sua cédula. Caso ele deseje, então a máquina imprime os segmentos 1,2 e

Algoritmo 1 Gerar o voto de um eleitor i

Input: $v_i = \{a_{i1}, a_{i2}, \cdots a_{iN}\}$, $publicKey = (p, g, \beta)$, identificação do eleitor ID_i

Output: A tupla $< ID_i, c_i = \{c_{i1}, \cdots, c_{iN}\}, S_i = \{S_{i1}, \cdots, S_{iN}\}, B_i = \{B_{i1}, \cdots, B_{iN}\}, X_i = \{X_{i1}, \cdots, X_{iN}\}, Y_i, H_i >$

1 **for** $j = \{1, \cdots, N\}$ **do**
2 $c[i,j] \leftarrow 0$
3 $s[i,j] \leftarrow 0$
4 $S[i,j] \leftarrow 0$
5 $B[i,j] \leftarrow 0$
6 $X[i,j] \leftarrow 0$
7 **for each** $a_{ij} \in v_i$ **do**
8 $r \leftarrow GerarAleatorio()$ *// r recebe um inteiro positivo aleatório* $\in Z_p$
9 $s[i,j] \leftarrow r$
10 $c[i,j] \leftarrow ElGamalExponencial(a_{ij}, publicKey)$
11 $S[i,j] \leftarrow ElGamal(s[i,j], publicKey)$
12 $B[i,j] \leftarrow BitCommitment(a_{ij}, c[i,j], publicKey)$
13 $X[i,j] \leftarrow PPK(c[i,j], publicKey, s[i,j], 1, 2)$
14 $SUM_2 \leftarrow c[i,1]$
15 $SUM_1 \leftarrow s[i,1]$
16 **for** $k = 2$ **to** N **do**
17 $SUM_1 \leftarrow SUM_1 + s[i,k]$
18 $SUM_2 \leftarrow SUM_2 \times c[i,k]$
19 $Y_i \leftarrow PPK(SUM_2, publicKey, SUM_1, N, N+1)$
20 $H_i \leftarrow Hash(B_i)$ *// Hash do vetor* B_i.
21 **return** $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$

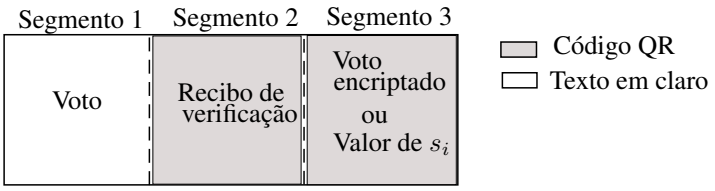


Figura 1. Formato de cédula impressa pela máquina de votação.

3, o segmento 3 contendo s_i em vez do voto encriptado. O eleitor dirige-se a uma máquina de verificação sem comunicação com a máquina de votação e apresenta os segmentos 2 e 3 para ela. A máquina de verificação executa o Algoritmo 2.

A máquina exibe o voto a_i', em texto em claro, reconstruído a partir do procedimento anterior. O eleitor confirma que a_i' é equivalente a seu voto em texto em claro a_i. Caso o eleitor detecte imprecisões entre o exibido pela máquina e o voto contido em texto em claro na sua cédula, ele começa todo processo novamente. Após auditar o seu voto, ele retorna à máquina de votação para gerar uma nova cédula, com um novo valor de s_i aleatório gerado pela máquina. Desta vez ele deve optar por não auditar a nova cédula para continuar o processo.

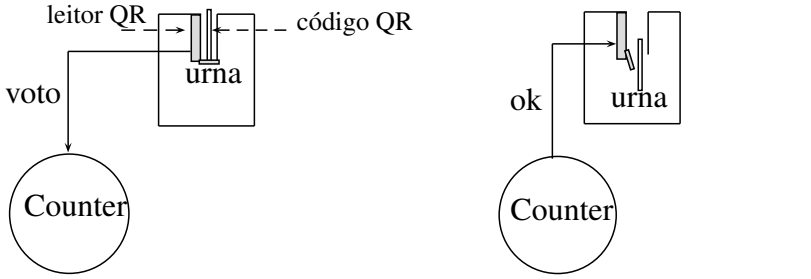
O eleitor destaca o segmento 2 contendo $< B_i >$ e leva-o como recibo. Depois, ele coloca o segmento 3 na região de leitores de código QR da urna física, como indicado na Figura 2, que lê e envia o conteúdo incluso no código QR, a tupla

Algoritmo 2 Audita o voto de um eleitor i

Input: $B_i = \{B_{i1}, \cdots, B_{iN}\}$, $s_i = \{s_{i1}, \cdots, s_{iN}\}$, $publickey(g, \beta, p)$
Output: O vetor a, representando B_i decifrado ou ABORT, em caso de B_i apresentar valores inválidos

```
1   for j = {1, ..., N} do
2       a[i, j] ← 0
3   for each B_ij ∈ B_i do
4       if B_ij == g^{s_ij} β^1 mod p
5           a[i, j] ← 1
6       else if B_ij == g^{s_ij} β^2 mod p
7           a[i, j] ← 2
8       else
9           return ABORT
10  return a
```

$< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$, para o *Counter*. O *Counter* ao receber uma tupla executa o Algoritmo 3. Assumimos que o *Counter* já foi previamente configurado com os valores da chave pública *publicKey*, e os vetores R e T.



a) Urna envia voto b) Counter contabiliza voto

Figura 2. Comunicação entre *counter* e Urna física.

Algoritmo 3 Verifica eligibilidade do eleitor i e integridade do seu voto para depois contabilizá-lo.

Input: Uma tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$, chave privada x, chave pública publicKey.
Output: *ACCEPT* ou *REJECT*

```
1   if Eleitor ID_i está irregular do
2       return REJECT
3   if PPK_Test(Y_i, publicKey) == FALSE do
4       return REJECT
5   for each voto cifrado c_ij ∈ c_i do
6       if PPK_Test(X_ij, publicKey) == FALSE do
7           return REJECT
8       AtualizaResultadoParcial(R_j, c_ij)          // R_j = R_j + c_ij
9       s_ij ← ElGamalDecifra(S_ij, x)
10      AtualizaVetorSecreto(T_j, s_ij)              // T_j = { T_j + s_ij   se T_ij ≠ ∅
                                                     //       { s_ij         caso contrário
11  return ACCEPT
```

Caso o $Counter$ aceite o voto, ele aciona a urna física para que ela aceite o voto físico. O eleitor deposita na urna o segmento superior da cédula, contendo o voto em texto claro, e destrói o segmento contendo $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$ em um local de descarte comum, com o mesário como testemunha.

4.4. Fase de Contagem de Votos

Nesta fase as k autoridades se reúnem a fim de reconstruírem o segredo x usado no processo de decifração da contagem de votos. O vetor $Resultado$ com a quantidade de votos de cada candidato é calculado pelo Algoritmo 4. Para obter o resultado é necessário subtrair o total de eleitores do número de votos de cada candidato para obter o resultado final.

Algoritmo 4 Computa o resultado da eleição

Input: Chave privada x, vetor com resultados criptografados $R = \{R_1 \cdots R_N\}$, quantidade de eleitores I
Output: $Resultado$- vetor com a quantidade de votos de cada candidato j

```
1   for each R_j ∈ R do
2       Resultado[j] ← 0
3   for each R_j ∈ R do
4       r_j ← ElGamalExponencialDecifra(R_j, x)
5       r_j ← r_j − I
6       Resultado[j] ← r_j
7   return Resultado
```

Após todos os eleitores terem votado, as autoridades publicam os totais de votos de cada candidato, $Resultado$, juntamente com os parâmetros de entrada do algoritmo 5 para que o eleitor possa verificar o total de votos de cada candidato.

Algoritmo 5 Algoritmo de verificação do total de votos de um candidato j pelo eleitor i.

Input: Chave pública $publicKey$, coluna da tabela pública $\{B_{1j}, B_{2j}, \cdots, B_{Ij}\}$,
candidato escolhido j, $Resultado = \{Resultado_1, Resultado_2, \cdots, Resultado_N\}$,
$T = \{T_1, T_2, \cdots, T_N\}$
Output: *TRUE*, se a verificação OK para o candidato j, *FALSE* caso contrário

```
1   B' ← B_1j                          // B_1j linha 1 coluna j da tabela publicada pelas autoridades.
2   for i = 2 to I do
3       B' ← B' × B_ij
4   B'' ← g^{T_j} β^{Resultado_j + N} mod p   // T_j = Σ_{i=1}^{I} s_ij e Resultado_j o total de votos do candidato j.
5   if B' == B'' do
6       return TRUE
7   else do
8       return FALSE
```

O sistema proposto também permite a contagem manual de votos a partir da urna física. Ela é indicada em casos de eventuais questionamentos sobre a integridade da eleição. O método manual é suposto ser seguro e transparente pois o leitor escolheu seu voto em segredo e pode conferir visualmente ele em texto em claro antes de depositá-lo na urna física.

Finalmente, cada eleitor $i = \{1, 2, 3, ..., n\}$ leva consigo um recibo contendo B_i. O sigilo total do voto é garantido uma vez que o eleitor i não conhece s_i, i.e., ele já foi destruído anteriormente, sendo impossível deduzir qualquer informação sobre a escolha do eleitor a partir de B_i, mesmo com a conivência do eleitor. Posteriormente, um eleitor interessado em confirmar que seu voto foi incluído na contagem pode verificar se o valor do *hash* (B_i) contido no seu recibo está incluso na tabela divulgada pelas autoridades.

5. Resultados e Discussão

5.1. Análise de Desempenho

A fim de validar o desempenho dos algoritmos calculamos os tempos de execução médio de cada um. Cada algoritmo foi executado 5 vezes. Os valores médios são mostrados nas Tabelas 3 e 4. O tempo é medido em segundos e o desvio padrão é indicado entre parênteses. Todos os testes foram realizados em um computador com processador Intel Quad Core i7-6500U CPU @ 2.50GHz e 8 GB de memória RAM, utilizando a linguagem de programação Python. O código-fonte da simulação utilizada para realizar os testes pode ser baixado no endereço `https://github.com/pdrvncs/UrnaPrototipo`.

A Tabela 3 apresenta os tempos de execução médio do Algoritmo 1, executado pela urna, e do Algoritmo 3, executado pelo *Counter*, para gerar um voto e processá-lo respectivamente, considerando um tamanho de chave em bits e uma quantidade de candidatos. Note que a escolha do candidato de um eleitor foi feita automaticamente pelo *software* de simulação selecionando um candidato aleatório. A Tabela 4 apresenta os tempos de execução médio para o Algoritmo 4 obter o resultado das eleições e o Algoritmo 5 fazer uma verificação individual considerando um tamanho de chave em bits e a quantidade de eleitores.

Os resultados dos experimentos indicam que os algoritmos possuem desempenho muito bom na prática, i.e, para computar o resultado de uma eleição (Algoritmo 4) com 1000 eleitores e tamanho de chave 1024 foram necessários menos de 10 segundos.

Tabela 3. Tempos de execução médio dos algoritmos de geração e de processamento do voto.

Tamanho da chave	Algoritmo 1		Algoritmo 3	
	Qtd de candidatos		Qtd de candidatos	
	100	1000	100	1000
512	0.82 (0.02)	7.87 (0.21)	0.75 (0.01)	7.36 (0.35)
1024	4.77 (0.03)	47.44 (0.48)	4.38 (0.18)	43.38 (1.56)

5.2. Análise e Discussão de Segurança do Sistema Proposto

Neste trabalho, assumimos que o problema do logaritmo discreto é difícil e que são seguidas todas as recomendações para a escolha dos parâmetros que o torna difícil. E que o esquema de comprometimento de bits tem sua segurança baseada na dificuldade deste problema. A seguir provamos que o esquema de votação proposto atende diversas propriedades de segurança descritas na Seção 2.

Tabela 4. Tempos de execução médio dos algoritmos de obtenção dos resultados e de verificação individual.

Tamanho da chave	Algoritmo 4		Algoritmo 5	
	Qtd de eleitores		Qtd de eleitores	
	100	1000	100	1000
512	0.09 (0.01)	0.91 (0.03)	0.01 (0.01)	0.68 (0.29)
1024	0.52 (0.01)	5.04 (0.04)	0.33 (0.01)	2.63 (0.03)

Teorema (Cada eleitor só pode votar uma vez)**.** *Se o eleitor vota mais de uma vez, então o* $Counter$ *registra somente o primeiro voto.*

Prova por contradição. Assuma que o eleitor vote mais de uma vez e o $Counter$ registre pelo menos dois destes votos. Mas, pelo Algoritmo 3, o *Counter* mantém o registro de todos os eleitores que podem votar e de todos os eleitores que já votaram impedindo que quem já votou vote novamente. Logo, o eleitor só vota uma única vez. □

Teorema (Todos os votos na urna, e somente esses, serão incluídos na contagem)**.** *Se a prova de conhecimento parcial é segura, então apenas votos válidos serão considerados na contagem.*

Prova por contradição. Assuma que a prova de conhecimento parcial é segura e um voto não válido seja considerado na contagem. Mas, pelo Algoritmo 3, o *Counter* realiza o teste de verificação da prova de conhecimento parcial, rejeitando qualquer voto com valores ilegais sem comprometer a privacidade do eleitor. Logo, apenas votos válidos são inclusos na contagem final. □

Teorema (O eleitor pode, após o voto, verificar se ele é realmente válido)**.** *Se o esquema de comprometimento de bits é seguro e a máquina de verificação é segura, então o eleitor pode verificar a validade do seu voto.*

Prova por contradição usando a contra-positiva. Se o eleitor não pode verificar a validade do seu voto, então o esquema de comprometimento de bits não é seguro e a máquina de verificação não é segura. No entanto, o eleitor abre o comprometimento B_i utilizando uma máquina de verificação sem comunicação com a máquina na qual votou e o esquema de comprometimento de bits é seguro pois se baseia no problema do logaritmo discreto. Logo, o eleitor pode, após o voto, verificar se ele é realmente válido. □

Teorema (Deve ser possível auditar a contagem)**.** *Se cada voto é válido, então a exatidão da contagem dos votos pode ser garantida.*

Prova por contradição. Assuma que cada voto é válido e a exatidão da contagem dos votos não pode ser garantida. Mas, pela propriedade homomórfica do ElGamal Exponencial, o produto de x números inteiros cifrados, decifra para a soma destes números. Logo, como os votos são cifrados através do ElGamal Exponencial e todos os votos são válidos, então o deciframento do resultado do produto de todos os votos cifrados é exatamente a soma de todos os votos. □

Teorema (O eleitor pode se convencer que seu voto realmente foi apurado)**.** *Se o voto do eleitor foi aceito pelo Counter, então o eleitor pode verificar que seu voto foi incluído na contagem.*

Prova direta. Assuma que o voto do eleitor foi aceito pelo *Counter*, como o eleitor leva consigo um recibo contendo a tupla $< ID_i, H_i, B_i >$ e as autoridades publicam uma tabela contendo todas as tuplas$< ID_i, H_i, B_i >$, $i = 1, 2, \cdots, n$, n o total de eleitores, aceitas pelo *Counter*. Então, todo eleitor em posse do seu recibo e com acesso à tabela pode conferir a inclusão do seu voto na contagem. □

Teorema (privacidade do voto incondicional)**.** *Se o esquema de comprometimento de bits é incondicionalmente seguro, então é impossível a dedução de qualquer informação sobre a escolha do eleitor, mesmo com a conivência deste, a partir dos dados publicados no canal público.*

Prova direta. Assuma que o esquema de comprometimento de bits seja incondicionalmente seguro, então os valores cifrados representando os votos dos eleitores divulgados no canal público gerados a partir do esquema são incondicionalmente seguros. □

Teorema (Ninguém deve conseguir alterar ou remover os votos na urna ou incluir votos ilegítimos)**.** *Se alguém altera o resultado da eleição, então qualquer eleitor pode mostrar que o resultado da eleição foi alterado.*

Prova por contradição. Assuma que alguém altere o resultado da eleição e que um eleitor não possa mostrar que o resultado da eleição foi alterado. Mas, a correção do resultado pode ser verificada pelo algoritmo 5. Logo, o eleitor pode mostrar que o resultado da eleição foi alterado. □

6. Conclusões

Neste trabalho, combinamos técnicas de encriptação homomórfica, esquemas de comprometimento e provas de conhecimento parcial para criar um esquema de votação seguro e transparente. Provou-se que o esquema reúne diversas propriedades de um esquema de votação considerado seguro e transparente, como proposto pela comunidade científica. Além disso, os resultados experimentais indicam a viabilidade de verificação do resultado da eleição por qualquer uma das partes externas (eleitor ou observador internacional, por exemplo).

Por um lado, a encriptação homomórfica com o ElGamal modificado resolveu de forma eficiente o problema da falta de confiança de partes externas no resultado da eleição divulgado pelas autoridades, já que qualquer parte interessada com acesso ao resultado publicado pode auditar o resultado da eleição. Por outro lado, o uso de esquemas de comprometimento habilitou qualquer eleitor a verificar a inclusão e a integridade de qualquer voto, sem comprometer o seu sigilo. No entanto, o aspecto de facilidade de uso do esquema considerando eleitores com pouca escolaridade ainda é uma questão em aberto.

Finalmente, é importante notar que apesar dos bons resultados experimentais quanto ao desempenho dos algoritmos, ainda estamos distantes de um sistema operável

em grande escala. Testes de escalabilidade do esquema e o seu impacto na transparência e na segurança de uma eleição não foram avaliados. Outras questões também precisam ser superadas, e.g., como garantir a inviolabilidade do software da urna? E analisar a complexidade do processo sob a perspectiva do eleitor. Por fim, também seria importante incorporar ao esquema técnicas de encriptação baseadas em reticulados [Pino et al. 2017] que são conjecturadas resistirem a ataques de computadores quânticos.

Referências

[Azougaghe et al. 2015] Azougaghe, A., Hedabou, M., and Belkasmi, M. (2015). An electronic voting system based on homomorphic encryption and prime numbers. In *2015 11th International Conference on Information Assurance and Security (IAS)*, pages 140–145.

[Brams and Fishburn 2005] Brams, S. J. and Fishburn, P. C. (2005). Going from theory to practice: the mixed success of approval voting. *Social Choice and Welfare*, 25(2):457–474.

[Graaf 2017] Graaf, J. A. M. V. d. (2017). Long-term threats to ballot privacy. *IEEE Security Privacy*, 15(3):40–47.

[Hirt and Sako 2000] Hirt, M. and Sako, K. (2000). Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'00, pages 539–556, Berlin, Heidelberg. Springer-Verlag.

[Lima et al. 2017] Lima, F. T., Gazziro, M. A., Batista Junior, A. A., Matias, P., and Costa, J. V. C. (2017). Urna eletrônica de terceira geração: Um protótipo para eleições auditáveis. In *XVII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais: SBSEG 2017*, pages 677–685, Brasília-DF.

[National Academies of Sciences and Medicine 2018] National Academies of Sciences, E. and Medicine (2018). *Securing the Vote: Protecting American Democracy*. The National Academies Press, Washington, DC.

[Peng and Bao 2011] Peng, K. and Bao, F. (2011). Efficient multiplicative homomorphic e-voting. In *Proceedings of the 13th International Conference on Information Security*, ISC'10, pages 381–393, Berlin, Heidelberg. Springer-Verlag.

[Pino et al. 2017] Pino, R. d., Lyubashevsky, V., Neven, G., and Seiler, G. (2017). Practical quantum-safe voting from lattices. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1565–1581, New York, NY, USA. ACM.

[Yang et al. 2018] Yang, X., Yi, X., Nepal, S., Kelarev, A., and Han, F. (2018). A secure verifiable ranked choice online voting system based on homomorphic encryption. *IEEE Access*, 6:20506–20519.