Transparência e Privacidade em Urnas Eletrônicas através de Encriptação Homomórfica e de Esquemas de Comprometimento

Pedro Vinícius Macêdo de Araújo

Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia - CCET
Departamento de Informática

São Luís
2018

Pedro Vinícius Macêdo de Araújo

Transparência e Privacidade em Urnas Eletrônicas através de Encriptação Homomórfica e de Esquemas de Comprometimento

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Antônio de Abreu Batista Júnior

São Luís
2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Araújo, Pedro Vinícius Macêdo de.
Transparência e privacidade em urnas eletrônicas através de encriptação homomórfica e esquemas de comprometimento / Pedro Vinícius Macêdo de Araújo. – 2018.
50 f.

Orientador(a): Antônio de Abreu Batista Júnior.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2018.

1. Criptografia. 2. Encriptação homomórfica. 3. Esquema de comprometimento. 4. Urna eletrônica. I. Batista Júnior, Antônio de Abreu. II. Título.

PEDRO VINÍCIUS MACÊDO DE ARAÚJO

TRANSPARÊNCIA E PRIVACIDADE EM URNAS ELETRÔNICAS ATRAVÉS DE ENCRIPTAÇÃO HOMOMÓRFICA E DE ESQUEMAS DE COMPROMETIMENTO

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

São Luís, 20 de dezembro de 2018.

BANCA EXAMINADORA

Prof. Antônio de Abreu Batista Júnior
Universidade Federal do Maranhão

Prof. Mário Alexandre Gazziro
Universidade Federal do ABC

Prof. Paulo Matias
Universidade Federal de São Carlos

Prof. Geraldo Braz Júnior
Universidade Federal do Maranhão

Agradecimentos

Agradeço a Deus, meus pais e meus professores, especialmente meu orientador.

Resumo

A democracia depende de uma ampla confiança na integridade das eleições. Em uma eleição que utilize sistemas eletrônicos, faz-se necessário que exista uma forma de satisfazer o requisito de transparência exigido para eleições públicas e o sigilo do voto garantido por lei. Neste trabalho, investigamos o uso de encriptação homomórfica e esquemas de comprometimento em um esquema de votação eletrônica com o objetivo de manter a privacidade dos eleitores e de permitir a auditação do processo. Além disso, nós apresentamos uma análise da sua segurança e de sua viabilidade de implementação, mostrando que o esquema proposto atende aos requisitos para uma eleição segura, justa e transparente.

Palavras-chave: Encriptação Homomórfica. Esquema de Comprometimento Incondicional. Votação Eletrônica.

Abstract

Democracy relies on broad confidence in the integrity of the voting process. In an e-voting system, there is a need to fulfill the required standards of transparency demanded by public elections and the privacy guaranteed by law. In this work, we investigate the use of homomorphic encryption and unconditional commitment schemes on an e-voting system aiming at maintaining voter's privacy and allowing auditioning of the process. Also, we present a security and feasibility analysis of the proposed scheme and it is shown that it meets the requirements for a safe, fair and transparent election.

Keywords: Homomorphic encryption. Unconditional commitment schemes. E-voting.

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

Atualmente, no Brasil, os eleitores precisam confiar na competência e boa-fé das autoridades eleitorais, pois não há uma forma de auditar o resultado das eleições. Se isso não bastasse, a segurança da urna eletrônica usada nas eleições é baseada na filosofia de "segurança por obscuridade", na qual acredita-se que o sigilo do funcionamento de um sistema pode tornar esse sistema mais seguro (GRAAF, 2017b). No entanto, é publicamente conhecido que esta prática pode esconder vulnerabilidades teóricas ou reais que podem ser descobertas mais cedo ou mais tarde. Recentemente, diversas falhas de segurança na urna eletrônica brasileira tem sido descobertas, como as relatadas em (MATIAS, 2018), reforçando a necessidade de formas alternativas de contagem de votos.

1.1 Motivação

Hoje, sabe-se que o uso de métodos eletrônicos de votação introduziu novos desafios. Antes, o eleitor votava em cédula de papel e, sigilosamente, escolhia o seu candidato. Agora, ele escolhe seu candidato em uma máquina e precisa confiar nela. De fato, com a eliminação das cédulas em papel a confiança na integridade da eleição ficou depositada exclusivamente no bom funcionamento da urna.

Recentemente, um número de trabalhos (LIMA et al., 2017; GRAAF, 2017a) embute o registro do voto digital em cédulas de papel por meio de códigos *QR*. O propósito principal destes trabalhos foi propor meios alternativos de auditar o resultado da eleição em caso de suspeita de fraude. Alguns outros trabalhos empregam protocolos de votação baseados em encriptação homomórfica (YANG et al., 2018; AZOUGAGHE; HEDABOU; BELKASMI, 2015; PENG et al., 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.

No entanto, apesar do grande esforço da comunidade científica em propror um sistema seguro e confiável nas últimas décadas, ainda não se tem um tal sistema de votação eletrônica em que o eleitor confie plenamente no resultado da eleição sem a necessidade

de confiar na competência e boa fé das pessoas. A seguir, apresentamos o objetivo deste trabalho.

1.2 Objetivos e Desafios da Pesquisa

Nosso objetivo é desenvolver um esquema de votação eletrônica baseado em encriptação homomórfica e esquemas de comprometimento que atenda ao modelo de segurança proposto pela comunidade científica. Esse objetivo impõe grandes desafios devido a necessidade de alcançar simultaneamente muitos requisitos aparentemente contraditórios como o sigilo do voto e a transparência requerida para eleições públicas.

Modelo de segurança

Aqui, consideramos que um esquema de votação eletrônica é seguro se ele satisfaz as seguintes propriedades:

1. (*Eligibility*) Somente eleitores válidos podem votar;
2. (*Unreusability*) Cada eleitor só pode votar uma vez;
3. (*Privacy and Anonymity*) Sigilo total do voto, impossibilitando a dedução de qualquer informação sobre a escolha do eleitor, mesmo com a conivência deste;
4. (*Individual Verifiability*) O eleitor pode, após o voto, verificar se ele é realmente válido;
5. (*Individual Verifiability*) 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. (*Fairness*) 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. (*Universal Verifiability*) Deve ser possível auditar a contagem.

O funcionamento de nossa proposta depende de alguns pressupostos e condições. Nós assumimos que:

1. Um adversário pode impedir um eleitor de votar e exigir informações sigilosas do eleitor. O adversário possui poder computacional limitado e só pode corromper uma parte das autoridades;

2. Uma fase de registro livre de adversários. Também assumimos que o eleitor realiza seu cadastro eleitoral sem a interferência de adversários.

3. O adversário 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 o momento de votação

4. Adversários não podem adulterar ou corromper os votos na urna.

5. O eleitor vota através de máquinas confiáveis.

O restante deste trabalho esta organizado como segue. Na seção 2, apresentamos uma descrição das primitivas criptográficas utilizadas e da notação que será utilizada neste trabalho, além de uma breve mostra de trabalhos correlatos que abordam o mesmo problema. Na seção 3, descrevemos passo a passo como a urna proposta funciona utilizando os conceitos expostos na seção anterior, acompanhado de algoritmos em pseudocódigo.

As seções 4 e 5 apresentam uma análise do funcionamento da urna. A seção 4 mostra como a urna alcança os requisitos do nosso modelo de segurança e a seção 5 apresenta uma investigação do desempenho computacional dos algoritmos propostos em resposta à diversos valores de entradas que possam impactar em suas respectivas performances.

Capítulo 2

Referencial Teórico e Trabalho Correlato

Este capítulo introduz os blocos de construção usados no desenvolvimento do esquema de votação proposto e faz-se uma análise crítica dos principais trabalhos correlatos.

2.1 Trabalho Correlato

Para alguns especialistas (GRAAF, 2017a; LIMA et al., 2017), a propriedade de dualidade, ou seja, a coexistência da votação eletrônica e em papel, é essencial para que não haja dúvidas da lisura do processo de votação. Em (GRAAF, 2017a) é proposto um sistema de votação baseado em encriptação homomórfica para realizar a contagem dos votos sem comprometer votos individuais, além de um esquema de comprometimento de bits incondicional para proteger o voto dos eleitores. Já (LIMA et al., 2017) propõe um protótipo de urna auditável, apresentando as propriedades de dualidade e obtenção do resultado através da leitura de *QR Codes*, onde partes interessadas podem acompanhar todo o processo de computação do resultado.

Outro sistema de votação eletrônica baseado em encriptação homomórfica é o proposto por (YANG et al., 2018). O sistema de e-voting (votação eletrônica) pela internet sugerido é inspirado no chamado *score voting*, um sistema onde os eleitores dão uma nota para cada candidato e o candidato com as maiores notas no final é o vencedor. Por ser um sistema de votação pela internet, os autores propõe uma série de medidas com o intuito de coibir fraudes provenientes de eleitores inválidos ou da corrupção do resultado final por votos ilegítimos. Cada eleitor autorizado deve gerar previamente uma chave de assinatura (garantindo que apenas eleitores válidos votem) e uma prova de conhecimento, responsável por garantir que os as cifras submetidas realmente representam votos válidos.

O nosso trabalho difere dos outros trabalhos mencionados anteriormente em diversas questões. Nós combinamos os pontos fortes destes trabalhos em um novo sistema de votação eletrônica. Nós adicionamos provas de conhecimento parcial ao esquema descrito

por (GRAAF, 2017a), além de incluir neste último um projeto de urna física onde a privacidade do voto é garantida. Nós também analisamos a segurança deste esquema.

2.2 Referencial Teórico

Nesta seção, introduziremos os principais conceitos das ferramentas e primitivas criptográficas utilizadas neste trabalho.

2.2.1 O problema do logaritmo discreto

O problema do logaritmo discreto é a base de vários protocolos de criptografia atuais, dentre eles o sistema ElGamal e o protocolo de comprometimento de bits utilizados neste trabalho. Esse problema é proveniente da dificuldade de computar o expoente de uma potência em um grupo multiplicativo, ou seja, encontrar um inteiro n que satisfaça a equação $a^n \equiv b \bmod p$, onde os inteiros a e b e um primo p são conhecidos. O problema do logaritmo discreto é considerado um problema "difícil", isto é, sem nenhum algoritmo de tempo polinomial que possa solucioná-lo (SMITH, 2005).

2.2.2 Criptografia de chave pública

A criptografia de chave pública é uma classe de criptossistemas que permite que duas partes compartilhem uma mensagem secreta, mantendo as propriedades de confidencialidade e integridade da mensagem. Esse sistema é dito de chave pública pois as partes concordam em um par de chaves: uma pública, usada para qualquer pessoa cifrar uma mensagem, e uma privada, mantida apenas por quem deve ler a mensagem. Tais protocolos são baseados na dificuldade de computar algumas funções em sentido "reverso", onde realizar alguma computação em sentido "direto"é trivial, mas reverter essa computação exige algoritmos com dificuldade computacional de tempo exponencial ou superpolinomial. Dentre essas funções, temos como exemplos o problema do logaritmo discreto, o problema da fatoração de inteiros e o cálculo de resíduos quadráticos (SMITH, 2005).

O acordo de chaves de Diffie-Hellman permite a implementação de um sistema de chave pública. Nesse protocolo, duas partes, Alice e Bob computam um inteiro de n bits representando a chave secreta x, usada para encriptar uma mensagem m que é compartilhada publicamente, mas que só pode ser decifrada utilizando a chave secreta. O acordo pode ser resumido como:

1. Escolhe-se um primo p grande (tamanho $n + 1$ bits) e um gerador $g \bmod p$.

2. Alice escolhe aleatoriamente um valor $a \bmod p$ e envia $g^a \bmod p$ para Bob, mantendo a secreto.

3. Bob escolhe aleatoriamente um valor b mod p e envia g^b mod p para Alice, mantendo b secreto.

4. Alice calcula $x = (g^b)^a \bmod p$ e Bob calcula $x = (g^a)^b \bmod p$, gerando a chave secreta x.

A segurança é proveniente do fato que não é possível encontrar a, b e x a partir dos valores compartilhados g^a e g^b sem computar um logaritmo discreto. Vários protocolos de criptografia de chave pública utilizam o acordo de chaves de Diffie-Hellman, dentre eles o criptossistema ElGamal, utilizado neste trabalho.

2.2.3 ElGamal

O ElGamal é um criptossistema de chave pública baseado no acordo de chaves de Diffie-Hellman, com sua segurança proveniente do problema do logaritmo discreto (ELGAMAL, 1985). 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.

2.2.3.1 Geração de chaves

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 como na Equação 1 e defina $\beta \equiv g^x \bmod p$

$$x = \sum_{i=1}^{k} y_i \bmod p \tag{1}$$

4. Defina (p, g, β) com a chave pública e x como a chave privada.

Cifração

2.2.3.2 ElGamal

Dado a chave pública (p, g, β) e uma mensagem m,

1. Escolha aleatoriamente $i \in \{\mathbb{Z}^*_{p-1} - 1\}$.

2. Calcule $(k_E, c) = (g^i, m\beta) \in \mathbb{Z}^*_p$. k_E é a chave efêmera.

ElGamal Exponencial

A fim de permitir a contagem dos votos, faz-se necessário o uso de um protocolo que apresente propriedade homomórfica. Um criptossistema de cifração $E()$ é dito homomórfico se para dados $E(x)$ e $E(y)$, pode-se obter $E(x \oplus y)$ sem descriptografar 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.

Dado a chave pública (p, g, β) e uma mensagem m,

1. Calcule $(k_E, c) = (g^i, g^m \beta) \in \mathbb{Z}_p^*$. k_E é a chave efêmera.

Decifração

ElGamal

1. Calcule $m = c(k_M)^{-1} \in \mathbb{Z}_p^*$, com k_M, a chave de mascaramento, tal que , $K_M \equiv (k_E)^x \bmod p$

ElGamal Exponencial

Dado a chave privada x, uma cifra c e sua chave efêmera k_E ,

1. Calcule $g^m = c(k_M)^{-1} \in \mathbb{Z}_p^*$, com k_M , a chave de mascaramento, tal que , $K_M \equiv (k_E)^x \bmod p$

2. Encontre m resolvendo o problema do logaritmo discreto $g^m \equiv c \bmod p$.

Devido à necessidade de computar um logaritmo discreto para encontrar m, o ElGamal Exponencial é viável apenas em situações em que é sabido que o valor de m não alcançará grandes valores.

Propriedade Homomórfica

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal Exponencial com uma mesma chave pública (p, g, β). Assim existe (k_{E1}, c_1) e (k_{E2}, c_2) tais que:

$$(k_{E1}, c_1) = (g^{i_1}, g^{m_1} \beta^{i_1}) \bmod p$$
$$(k_{E2}, c_2) = (g^{i_2}, g^{m_2} \beta^{i_2}) \bmod p$$

Logo,

$$\begin{aligned}(k_{E1}, c_1) \times (k_{E2}, c_2) &= (g^{i_1}, g^{m_1}\beta^{i_1}) \times (g^{i_2}, g^{m_2}\beta^{i_1}) \bmod p \\ &= (g^{i_1} \times g^{i_2}, g^{m_1}g^{m_2} \times \beta^{i_1}\beta^{i_2}) \bmod p \\ &= (g^{i_1+i_2}, g^{m_1+m_2} \times \beta^{i_1+i_2}) \bmod p \\ &= (g^{i}, g^{m} \times \beta^{i}) \bmod p \end{aligned} \quad (2)$$

em que $m = m_1 + m_2$ e $i = i_1 + i_2$. Nota-se que esse sistema não apresenta privacidade incondicional: basta ter acesso à chave privada para revelar o conteúdo de uma cifra. Sendo assim, é necessário o uso de uma primitiva criptográfica que garanta o sigilo incondicional dos votos dos eleitores. Para satisfazer esse requisito, utilizaremos um esquema de comprometimento de bits incondicional.

2.2.4 Esquema de comprometimento de bits incondicional

Um protocolo de comprometimento é uma primitiva criptográfica que permite 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. Além disso, o protocolo não deve permitir que o emissor mude de ideia e revele um valor diferente do comprometido. Esse esquema é basicamente composto de duas fases. Na primeira, o comprometimento, o emissor compromete-se com um valor e envia para o receptor uma evidência b, representando o comprometimento com o valor escolhido. O receptor não deve ser capaz de obter nenhuma informação sobre o valor comprometido através da evidência b. Na segunda fase, a revelação, o emissor pode opcionalmente, revelar o valor escolhido para o receptor, tendo que convencê-lo que o valor revelado é realmente o valor comprometido original (DAMGARD; NIELSEN,).

Tais esquemas possuem as propriedades de ocultação e de amarração, responsáveis por proteger os interesses do emissor e do receptor, respectivamente. Através da ocultação, ninguém além do emissor é capaz de obter o valor comprometido a partir da evidência publicada b. Por outro lado, o receptor possui a certeza que o emissor não pode revelar um valor diferente do valor comprometido original. Esses esquemas são divididos em duas classes:

- **Incondicionalmente seguro para o receptor e computacionalmente seguro para o emissor**: Nesse cenário, mesmo com poder computacional infinito um emissor não poderá revelar um valor diferente do comprometido. Já um receptor com poder computacional polinomial não conseguirá descobrir o valor ocultado facilmente.

- **Incondicionalmente seguro para o emissor e computacionalmente seguro para o receptor**: Um comprometimento b não revela nenhuma informação sobre o valor escondido, mesmo para um receptor com poder computacional infinito.

Por outro lado, um emissor com poder computacional "muito grande"possui uma pequena chance de revelar um valor diferente do comprometido original.

Com o ElGamal, basta a chave privada para descriptografar uma mensagem cifrada c. A fim de proteger os votos dos eleitores, será usado um esquema de comprometimento de bits de segurança incondicional. Dada uma mensagem m e um valor $s \in \{\mathbb{Z}_{p-1}^* - 1\}$ escolhido aleatoriamente, o *bit-commitment* da mensagem m é dado pela expressão:

$$BitCommitment(m, s) = g^s \beta^m \bmod p \tag{3}$$

Teorema 1 (Amarração)**.** *Se o problema do logaritmo discreto é seguro, então não é possível alterar os valores comprometidos pelo protocolo de comprometimento de bits incondicional.*

Seja um *bit commitment* b, dado por $b = g^s \beta^m$. Suponhamos que é possível alterar o valor m comprometido, ou seja, encontrar s' e m' tais que $g^s \beta^m = g^{s'} \beta^{m'}$. Isso implica que é possível calcular:

$$\begin{aligned} g^s \beta^m &= g^{s'} \beta^{m'} \\ g^{(s-s')} &= \beta^{(m'-m)} \\ g^{(s-s')/(m'-m)} &= \beta \end{aligned} \tag{4}$$

Assim, partindo do princípio que o problema do logarito discreto é seguro, ocorre uma contradição.

Esse esquema garante privacidade incondicional, pois dado um *Bit Commitment* C, não é possível obter informação sobre o valor de m, já que cada valor possível tem a mesma probabilidade de ser o correto, desde que s e os parâmetros g e β tenham sido gerados de forma adequada.

Além disso, esse esquema de comprometimento possui propriedade homormófica, o que o torna ideal para os propósitos deste trabalho. Sejam bc_1 e bc_2 duas cifras resultantes do *bit commitment* com uma mesma chave pública (p, g, β). Assim existe bc_1 e bc_2 tais que:

$$\begin{aligned} bc_1 &= g^{s_1} \beta^{m_1} \bmod p \\ bc_2 &= g^{s_2} \beta^{m_2} \bmod p \end{aligned}$$

Logo,

$$\begin{aligned} bc_1 \times bc_2 &= (g^{s_1}\beta^{m_1} \bmod p) \times (g^{s_2}\beta^{m_2} \bmod p) \\ &= g^{s_1}g^{s_2} \times \beta^{m_1}\beta^{m_2} \bmod p \\ &= g^{s_1+s_2} \times \beta^{m_1+m_2} \bmod p \\ &= g^{s} \times \beta^{m} \bmod p \end{aligned} \tag{5}$$

em que $m = m_1+m_2$ e $s = s_1+s_2$. O fato de que cifras são utilizadas no nosso protocolo implica que eventuais valores inválidos podem ser cifrados, tornando-se indistinguíveis de valores válidos e prejudicando o resultado do processo. A fim de solucionar esse problema, utilizaremos uma forma de garantir que as cifras equivalem à valores válidos, mas sem comprometer o sigilo dos votos.

2.2.5 Prova de Conhecimento Parcial

Prova de conhecimento parcial é um método de garantir 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 com valores válidos, mas sem comprometer o sigilo dos votos encriptados.

A seguir damos o procedimento para geração da prova e como é feita a verificação (YANG et al., 2018).

Prova

1. Seja uma mensagem m_1 cifrada utilizando ElGamal exponencial $(k_E, c) = (g^i, g^{m_1}\beta^i) \bmod p$
2. Gera-se números aleatórios $t, v_2, s_2 \in Z_p$
3. Computa-se $T_0 = g^t$
4. Computa-se $T_1 = \beta^t$
5. Computa-se $T_2 = (g^{m_2 v_2}\beta^{s_2})/c^{v_2}$
6. Computa-se $v = hash(k_E||c||T_0||T_1||T_2)$
7. Computa-se $v_1 = v_1 XOR v_2$
8. Computa-se $s_1 = iv_1 + t$
9. Envia $k_E, c, T_0, T_1, T_2, v_1, v_2, s_1, s_2$ para o verificador

Verificação

1. Verificar se $v_1 XOR v_2 = hash(k_E||c||T_0||T_1||T_2)$
2. Verificar se $g^{s_1} = T_0 k_E^{v_1}$
3. Verificar se $\beta^{s_1} = T_1(c/g^{m_1})^{v_1}$
4. Verificar se $\beta^{s_2} = T_2(c/g^{m_2})^{v_2}$

Se todos os testes passarem, a mensagem cifrada pode ser considerada uma encriptação de m_1 ou m_2. Assim, o verificador pode confirmar que o voto é válido sem comprometimento do sigilo do voto.

2.2.6 Assinatura Digital

Em um protocolo de criptografia de chave pública, qualquer pessoa que tenha acesso à chave pública pode cifrar e transmitir uma mensagem, ou seja, adversários podem assumir uma identidade e enviar cifras afirmando ser quem não são. Uma das formas de sanar esse problema é por meio do uso de assinaturas digitais. Existem três usos básicos para uma assinatura digital:

1. Garantir a autenticidade, não permitindo que um adversário possa enviar uma mensagem cifrada fazendo-se passar por outra pessoa.
2. Garantir integridade, ou seja, que uma dada mensagem não foi alterada durante a transmissão.
3. Não-repúdio, impedindo que um emissor negue ter enviado uma mensagem após tê-lo feito.

Uma assinatura digital é um valor dependente de algum segredo que só é conhecido por quem assina e que, necessariamente, deve ser passível de verificação. Uma parte neutra deve poder resolver situações de conflito provenientes de adversários enviando mensagens em nome de outros e de emissores negando terem enviado uma mensagem, sem precisar de acesso aos valores secretos acordados (**??**).

Seja H uma função de *hash* resistente a colisões. Para assinar uma mensagem m utilizando uma chave pública (p, g, β) e uma chave privada x , computa-se:

1. Um valor aleatório k tal que $0 < k < p$ com k e $p - 1$ coprimos.
2. $r = g^k \bmod p$
3. $s = (H(m) - xr)k^{-1} \bmod (p - 1)$

O par (r, s) é a assinatura da mensagem m.

2.3 Notação

Nesta seção apresentamos a notação usada neste trabalho. Essa notação é usada por todos os algoritmos que descreveremos nas próximas seções.

- ID_i: Identificação do eleitor i.
- $publicKey$: Chave pública composta pelos parâmetros (p, g, β).
- N: Quantidade de candidatos elegíveis.
- I: Quantidade de eleitores registrados.
- $v_i = \{a_{i1}, a_{i2}, \cdots a_{iN}\}$: Voto do eleitor i como definido em (6).
- $c_i = \{c_{i1}, c_{i2}, \cdots c_{iN}\}$: v_i cifrado através do ElGamal Exponencial, onde cada elemento $c_{ij} \in c_i$ é definido como:

$$c_{ij} = ElGamalExponencial(a_{ij}, publicKey)$$

- $s_i = \{s_{i1}, s_{i2}, \cdots s_{iN}\}$: vetor de valores aleatórios escolhidos pela urna, com $s_{ij} \in Z_p$.
- $B_i = \{B_{i1}, B_{i2}, \cdots B_{iN}\}$: v_i cifrado através do esquema de comprometimento de bits incondicional e do vetor secreto s_i, onde cada elemento $B_{ij} \in B_i$ é definido como:

$$B_{ij} = BitCommitment(a_{ij}, s_{ij}, publicKey)$$

- $S_i = \{S_{i1}, S_{i2}, \cdots S_{iN}\}$: s_i cifrado através do ElGamal , onde cada elemento $s_{ij} \in S_i$ é definido como:

$$S_{ij} = ElGamal(s_{ij}, publicKey)$$

- $X_i = \{X_{i1}, X_{i2}, \cdots X_{iN}\}$: Prova de conhecimento parcial para o valor $c_{ij} \in c_i$, onde cada elemento $X_{ij} \in X_i$ é definido como:

$$X_{ij} = PPK(c_{ij}, publicKey, 1, 2)$$

- Y_i: Prova de conhecimento parcial para o produto dos valores $c_{ij} \in c_i$, definida como:

$$Y_i = PPK(SUM_{c_i}, publicKey, N, N+1)$$

  Em que SUM_{c_i} é a soma .. Devido à propriedade homomórfica do ElGamal Exponencial, o produto dos valores $c_{ij} \in c_i$ decifrados resulta na somatória dos valores em texto puro de $c_{ij} \in c_i$.

- $ElGamalExponencial(m, publicKey)$: Cifragem do valor m usando a chave pública $publicKey$, através do método de encriptação ElGamal Exponencial.

- $ElGamalExponencialDecifra(c, publicKey, x)$: Função que recebe uma cifra c do ElGamal Exponencial, a chave pública $publicKey$ e a chave privada x e decifra c.
- $ElGamal(m, publicKey)$: Cifragem do valor m usando a chave pública $publicKey$, através do método de encriptação ElGamal.
- $ElGamalDecifra(c, publicKey, x)$: Função que recebe uma cifra c do ElGamal tradicional, a chave pública $publicKey$ e a chave privada x e decifra c.
- $BitCommitment(m, s, publicKey)$: Esquema de comprometimento de bits incondicional para uma mensagem m, ocultado pelo valor secreto s e a chave pública $publicKey$.
- $PPK(c, publicKey, s, m_1, m_2)$: Prova de conhecimento parcial para um valor cifrado c, utilizando a chave pública $publicKey$ e o valor secreto s. A prova mostra que c decifrado equivale a m_1 ou m_2.
- $PPK_{Test}(z, publicKey)$: Algoritmo que verifica se uma prova de conhecimento parcial z é válida, retorna TRUE se os testes passarem, FALSE, caso contrário.
- $Sign(m, x, publicKey)$: Assinatura de uma mensagem m usando a chave secreta x e a chave pública $publicKey$.
- $R = \{R_1, R_2, \cdots R_N\}$: vetor contendo o resultado parcial cifrado através do ElGamal exponencial para cada candidato $R_i \in R$.
- $AtualizaResultadoParcial(R_{ij}, c_{ij})$: Função que atualiza o resultado parcial cifrado de um candidato $R_j \in R$ recebendo um novo valor cifrado pelo ElGamal exponencial $c_{ij} \in c_i$, definida como:

$$R_j = R_j \times c_{ij}$$

- $T = \{T_1, T_2, \cdots T_N\}$: vetor contendo o somatório dos valores secretos s_{ij} de cada candidato.
- $AtualizaVetorSecreto(T_j, s_{ij})$: Função que atualiza o valor secreto correspondente a um candidato $T_j \in T_i$ recebendo um novo valor $s_{ij} \in s_i$, definida como:

$$T_j = \begin{cases} T_j + s_{ij} & se\ T_{ij} \neq \emptyset \\ s_{ij} & \text{caso contrário} \end{cases}$$

- $k = \{k_1, k_2, \cdots k_I\}$: vetor divulgado em forma de tabela após a eleição para fins de auditação, onde cada elemento $k_i \in k$ é uma tupla contendo $< ID_i, B_i, H_i >$.

Capítulo 3

Esquema de Votação Eletrônica Proposto

Levando em consideração o modelo de segurança e os blocos de construção introduzidos, apresentaremos agora o nosso esquema de votação eletrônica baseado em encriptação homomórfica e esquema de comprometimento de bits incondicional. O esquema é composto de quatro fases e envolve três participantes:

1. Eleitores
2. Autoridades
3. Counter

O counter é responsável por contar os votos. A Tabela 1 mostra as estruturas de dados que são mantidas pelo *Counter* para controle da contagem de votos.

3.1 Fase de Configuração da Eleição

Essa fase engloba dois procedimentos principais, o primeiro deles é a distribuição das chaves privadas usadas pelas autoridades para publicação do resultado final da eleição, e o segundo é a configuração do *Counter* para receber somente votos de eleitores aptos a votar lidos a partir de leitores de códigos QR acoplados nas urnas físicas e enviados para ele. Além disso, a estrutura de dados representando a tabela que será publicada após o resultado da eleição é configurada, a fim de receber os dados necessários (como a chave pública) para auditação posterior.

3.2 Fase de Registro do Eleitor

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

Tabela 1 – Estruturas de dados mantidas pelo *Counter*

	Descrição	Utilidade
R	Vetor contendo o resultado parcial da eleição. Cada elemento $R_j \in R$ representa o resultado para o candidato j.	Contém o resultado parcial cifrado para cada candidato, atualizado toda vez que o *Counter* recebe um voto. A fim de não comprometer o andamento da eleição, deve ser mantido cifrado até o momento da contagem dos votos.
T	Vetor que contem o somatório dos valores secretos s_{ij} de cada voto, obtidos decifrando S_i ao receber um novo voto.	Deve ser mantido para fins de auditação posterior. Devido ao fato de S_i ser encriptado através do ElGamal tradicional, sem propriedade homomórfica, é necessário que S_i seja descifrado ao ser recebido e utilizado para atualizar T.

permissão para votar.

3.3 Fase de Votação

Nessa etapa, o eleitor gera seu voto, que é processado pelo *Counter*. A Tabela 2 apresenta os valores que são recebidos pelo *Counter* para cada voto e sua utilidade.

1. No esquema proposto, o voto v_i do eleitor i em seu candidato j é representado por uma linha da matriz $I \times N$ $A = (a_{ij})$, $v_i = \{a_{i1}, a_{i2}, \cdots a_{iN}\}$. Em que I é o número de eleitores e N o número de candidatos, e cada elemento da matriz é definido como segue:

$$a_{ij} = \begin{cases} 2 & \textit{se } i \textit{ votou em } j \\ 1 & \text{caso contrário} \end{cases} \tag{6}$$

   Através dessa definição, um voto pode ser considerado válido se para cada $a_{ij} \in v_i$ temos:

$$a_{ij} = \begin{cases} 2 & \textit{se } i \textit{ votou em } j \\ 1 & \text{caso contrário} \end{cases} \tag{7}$$

$$\sum_{j=1}^{N} a_{ij} = \begin{cases} N+1 & \text{se } i \text{ escolheu um candidato} \\ N & \text{se } i \text{ votou nulo} \end{cases} \tag{8}$$

   Assume-se que a máquina codifica o voto do eleitor em um vetor v_i conforme definido em (6). Quando o eleitor confirma a sua escolha de candidado, a máquina deve realizar os procedimentos descritos no Algoritmo 1.

Tabela 2 – Valores processados pelo *Counter*

	Descrição	Utilidade
ID_i	Identificação do eleitor.	Utilizado pelo *Counter* para confirmar a eligibilidade do eleitor.
c_i	A escolha do eleitor cifrada através do ElGamal Exponencial.	Decifrado utilizando a chave privada x. Deve ser recebido pelo *Counter*, utilizado para atualizar o resultado parcial R_i e descartado, a fim de não comprometer a privacidade dos eleitores.
S_i	Vetor s_i cifrado através do ElGamal tradicional, utilizado na cifração de B_i.	Decifrado utilizando a chave privada x. Deve ser recebido pelo *Counter*, utilizado para atualizar T_i e descartado, a fim de não comprometer a privacidade dos eleitores.
B_i	Esquema BitCommitment para o voto, publicado no canal público ao final da eleição.	Usado para fins de confirmação que seu voto foi incluído na contagem (eleitores) ou auditação do resultado (outras partes). É incondicionalmente seguro enquanto o valor s_i utilizado na cifração não for comprometido.
X_i	Prova que $c_{ij} \in c_i$ representa um valor válido.	Usado pelo *Counter* para confirmar que o voto submetido é válido.
Y_i	Prova que c_i representa um valor válido.	Usado pelo *Counter* para confirmar que o voto submetido é válido.
H_i	Hash de B_i.	Utilizado para facilitar a consulta ao canal público.

Conhecendo o vetor s_i usado na cifração do *Bit Commitment*, é trivial descobrir a opção de um eleitor i para o candidato j, basta testar qual valor de a_{ij} (1 ou 2) satisfaz a equação $B_{ij} = g^{s_{ij}}\beta^{a_{ij}} \bmod p$. Já o contrário, provar que o eleitor votou ou não em um candidato apenas com o valor a_{ij}, é impraticável se o vetor s_i foi gerado de forma adequada (propriedade garantida pela segurança incondicional do esquema de comprometimento). Isso impede que se extraia qualquer informação do voto do eleitor, tanto por terceiros como pelo próprio eleitor.

2. Em seguida, a urna imprime uma cédula como na Figura 1. A cédula é dobrável nas três regiões tracejadas. O primeiro *QR Code* contém $< B_i >$ e o segundo contém a tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$.

3. Após a impressão, o eleitor opcionalmente pode auditar sua cédula. Se escolher pela auditação, a máquina de votação imprime um novo *QR Code*, dessa vez contendo s_i e reinicia o processo. O eleitor dirige-se à uma máquina de verificação sem comunicação com a máquina de votação. O eleitor então mostra o novo *QR Code* e o *QR Code* da sua cédula que contém $< B_i >$. A máquina executa o Algoritmo 2.

   A máquina exibe o voto a'_i, em texto em claro, reconstruído a partir do procedimento

Alg. 1 Vote casting

Input: $v_i = \{a_{i1}, a_{i2}, \cdots a_{iN}\}$, chave pública $publicKey$, identificação do eleitor ID_i.
Output: A tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$

$c_i \leftarrow \emptyset$
$s_i \leftarrow \emptyset$
$S_i \leftarrow \emptyset$
$B_i \leftarrow \emptyset$
$X_i \leftarrow \emptyset$
for each $a \in v_i$ **do**
    $x \leftarrow h$ // *h é um inteiro positivo aleatorio* $\in Z_p$
    $s_i \leftarrow s_i \cup x$
    $c \leftarrow ElGamalExponencial(a, publicKey)$
    $c_i \leftarrow c_i \cup c$
    $S_i \leftarrow S_i \cup ElGamal(x, publicKey)$
    $B_i \leftarrow B_i \cup BitCommitment(a, c, publicKey)$
    $X_i \leftarrow X_i \cup PPK(c, publicKey, x, 1, 2)$
$SUM_{c_i} \leftarrow c_{i1}$
$SUM_{s_i} \leftarrow s_{i1}$
for $k = 2$ **to** N **do**
    $SUM_{s_i} \leftarrow SUM_{s_i} + s_{ik}$
    $\sigma \leftarrow c_{ik} \times SUM_{c_i}$
    $SUM_{c_i} \leftarrow \sigma$
$Y_i \leftarrow PPK(SUM_{c_i}, publicKey, SUM_{s_i}N, N + 1)$
$H_i \leftarrow Hash(B_i)$
return $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$

Alg. 2 Vote verfication

Input: B_i, $s_i, publicKey$
Output: O vetor a'_i, representando B_i decifrado ou ABORT, em caso de B_i apresentar valores inválidos

$a'_i \leftarrow \emptyset$
for each $B_{ij} \in B_i$ **do**
    if $B_{ij} == g^{s_{ij}}\beta^1 \mod p$
        $a'_i \leftarrow a'_i \cup 1$
    else if $B_{ij} == g^{s_{ij}}\beta^2 \mod p$
        $a'_i \leftarrow a'_i \cup 2$
    else
        return $ABORT$
return a'_i

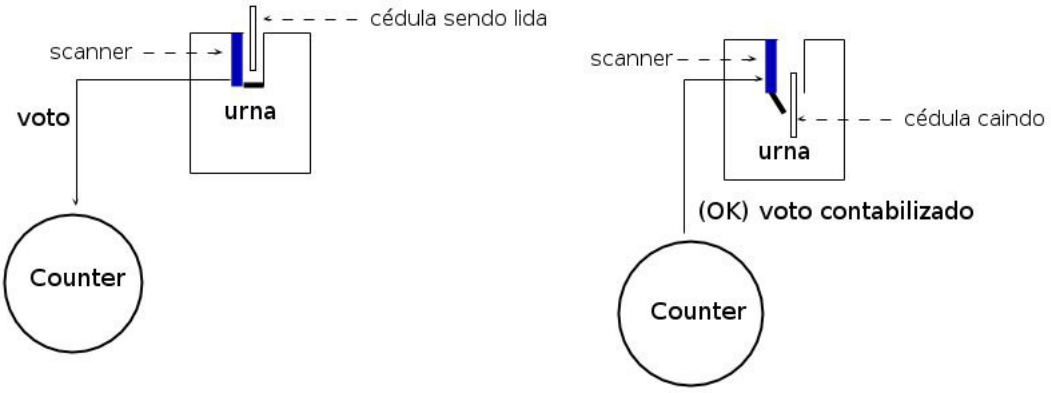
anterior. O eleitor confirma que a'_i é equivalente à seu voto a_i. Caso o eleitor detecte divergência entre o exibido pela máquina e o voto contido em texto em claro na sua

Figura 1 – Exemplo de cédula

cédula, ele começa todo processo novamente. Caso o eleitor aceite a auditação do seu voto, ele retorna à máquina de votação para gerar uma nova cédula, com um novo valor s_i.

4. O eleitor destaca o trecho da cédula contendo H_i e $< B_i >$ e o leva como recibo. Depois, ele coloca a outra parte da cédula na região de leitores de QR *code* da urna física, como mostrado na Figura 2, que lê e envia o conteúdo incluso no código QR, a tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$, para o *Counter*. O *Counter* ao receber uma tupla, deve:

Figura 2 – Urna física com leitores de códigos QR.



❏ Confirmar a eligibilidade do eleitor através do seu ID_i, verificando se este está

devidamente registrado e que ainda não votou. Rejeitar o voto em caso de falha.

- ❏ Realizar os testes da prova de conhecimento parcial para o voto. Rejeitar em caso de falha.
- ❏ Atualizar R, o vetor que representa o resultado parcial para cada candidato $R_i \in R$. Esse vetor é mantido encriptado até o momento da contagem dos votos.
- ❏ Atualizar T, o vetor que representa o somatório parcial do vetor s_i de cada voto. Devido à ser impraticável decifrar grandes valores através do ElGamal Exponencial, é necessário que s_i seja cifrado utilizando o ElGamal clássico (representado pelo vetor S_i), o que, devido à inexistência de propriedade homomórfica aditiva, implica que cada vetor S_i deve ser decifrado ao ser recebido.
- ❏ Descartar os valores c_i e S_i recebidos após processar as operações.
- ❏ Registrar a tupla $< ID_i, H_i, B_i >$ na tabela k.

Esse passo é mostrado em forma de pseudocódigo no Algoritmo 3.

Alg. 3 Update results

Input: Uma tupla $< ID_i, c_i, S_i, B_i, X_i, Y_i, H_i >$, chave privada x.
Output: *ACCEPT*, representando voto aceito, *REJECT*, representando voto rejeitado.

```
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       R ← R ∪ AtualizaResultadoParcial(R_j, c_ij)
9       s_ij ← ElGamalDecifra(S_ij, publicKey, x)
10      T ← T ∪ AtualizaVetorSecreto(T_j, s_ij)
11  return ACCEPT
```

5. 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.

3.4 Fase de Contagem de Votos

Nesta fase as k autoridades se reúnem a fim de reconstruírem a chave secreta x usada no processo de decifração da contagem total. O total de votos $Resultado_j$ de cada candidato j é calculado pelo Algoritmo 4. Como as opções possíveis para os candidatos são 1 ou 2, é necessário subtrair a quantidade de candidatos possíveis de cada valor decifrado para obter o resultado total.

Alg. 4 Obtain election results

Input: R_i, $publicKey$,x
Output: $Resultado$, vetor com a quantidade de votos de cada candidato j

```
1   Resultado ← ∅
2   for each R_j ∈ R do
3       r_j ← ElGamalExponencialDecifra(R_j, publicKey, x)
4       r_j ← r_j − N
5       Resultado ← Resultado ∪ r_j
6   return Resultado
```

Depois, elas publicam o resultado final. As autoridades então assinam $Resultado_j$ e o publicam, acompanhado da assinatura $\jmath_{Autorities}$. Para fins de auditação posterior, as autoridades devem publicar T_j, $publicKey$ e x.

$$\jmath_{Autorities}(r, s) = Sign(Resultado, x, publicKey) \tag{9}$$

Conforme ilustrado na Tabela 3, k é divulgado como uma tabela contendo o *bit-commitment* de cada eletor i, acompanhado de sua identificação e do *hash* da cédula de recibo.

Tabela 3 – Exemplo de tabela divulgada no canal público

Eleitor	Hash	Bit-Commitment
ID_1	H_1	B_1
ID_2	H_2	B_2
...	...	...
ID_I	H_I	B_I

Opcionalmente, é possível realizar a contagem manual das cédulas em papel, para confirmar o resultado e auditar a integridade do processo.

Capítulo 4

Análise de Segurança

Nesse capítulo mostramos que a urna proposta atende aos requisitos definidos em nosso modelo de segurança e a transparência exigida para eleições públicas.

4.1 Auditação individual

No momento da votação, o eleitor pode opcionalmente escolher auditar o voto na sua cédula, com o objetivo de comprovar que o valor no *QR Code* da sua cédula equivale ao seu voto. Utilizando o vetor secreto s, a máquina de verificação testa qual dos valores possíveis (1 ou 2) abre cada *bit commitment* no vetor B. Assume-se que a máquina faz o processo reverso da função definida em 6, exibindo o número do candidato escolhido através do vetor gerado após a abertura de B.

4.2 Contagem manual das cédulas

O fato do sistema proposto produzir votos eletrônicos e em papel permite que o resultado seja comprovado através de duas formas diferentes e independentes. As cédulas em papel são evidência física do resultado da eleição, podendo servir como forma de auditação secundária da contagem eletrônica.

4.3 Auditação do resultado

Alguma parte interessada em auditar o resultado da eleição primeiramente deve confirmar a validade da assinatura $\jmath_{Autorities}(r, s)$ das autoridades, através do teste da Equação 10, com $M = Resultado_i$. Isso evita que um atacante publique uma tabela k se passando pelas autoridades.

$$g^{H(M)} \equiv \beta^{r_M} r^{s_M} \bmod p \tag{10}$$

Depois, tal parte pode conferir a tabela k publicada pelas autoridades e realizar o teste da Equação 11 para um candidato escolhido j, :

$$\prod_{i=1}^{I} B_{ij} \equiv g^{T_j} \beta^{Resultado_j + N} \bmod p \tag{11}$$

Onde B corresponde ao vetor coluna de *bit commitments* da tabela divulgada pelas autoridades e T_j e $Resultado_j$ são os valores divulgados pelas autoridades para o candidato j. Em caso de fraude nos resultados ou na tabela divulgada, o teste da equação 11 falhará. O Algoritmo 5 apresenta o processo de auditação para um candidato j a partir dos dados divulgados pelas autoridades.

Alg. 5 Algoritmo de auditação

Input: Chave pública $publicKey$, vetor coluna da tabela pública $B = \{B_1, B_2, \cdots, B_I\}$, candidato escolhido j, $Resultado = \{Resultado_1, Resultado_2, \cdots, Resultado_N\}$, $T = \{T_1, T_2, \cdots, T_N\}$
Output: *TRUE*, se a auditação para o candidato j passar, *FALSE* caso contrário

```
1   B' ← B_1j
2   for i = 2 to I do
3       B' ← B' × B_ij
4   B'' ← g^{T_j} β^{Resultado_j + N} mod p
5   if B' == B'' do
6       return TRUE
7   else do
8       return FALSE
```

4.4 Verificação da inclusão do voto do eleitor i na contagem total

Cada eleitor $i = \{1, 2, 3, ..., n\}$ leva consigo um recibo contendo B_i calculado pela Equação 3 . O sigilo total do voto é garantido uma vez que o eleitor i não conhece s_i, sendo impossível deduzir qualquer informação sobre a escolha do eleitor, mesmo com a conivência deste, a partir de B_i.

Um eleitor interessado em confirmar que seu voto foi incluído na contagem pode:

Step 1 - Verifica que o valor do *hash* h'_i contido no seu recibo está incluso na tabela divulgada pelas autoridades.

Step 2 - Lê o valor B_i contido no *QR Code* e verifica se é equivalente à entrada B'_i que acompanha h_i na tabela.

4.5 Provando que a urna atende ao modelo de urna segura

Teorema 2 (Uniqueness)**.** *Se o eleitor vota mais de uma vez, então o Counter resgista somente o primeiro voto.*

O *Counter* mantém um registro de todos os eleitores que podem votar e de todos os eleitores que já votaram, como mostrado pelo Algoritmo 3. Além disso, a eligibilidade do eleitor pode ser confirmada pelo mesário presente na seção através da assinatura ou registro da presença do eleitor. Logo, o *Counter* detecta e rejeita votos repetidos oriundos de um mesmo eleitor.

Teorema 3 (ppk)**.** *Se a prova de conhecimento parcial é segura, então apenas votos válidos serão considerados na contagem.*

Assuma que a prova de conhecimento parcial é segura. Pela definição da prova de conhecimento parcial, a prova permite que um receptor de uma mensagem cifrada c possa comprovar que o valor ocultado realmente equivale à uma de duas entradas possíveis m_1 ou m_2, sem comprometer nenhuma informação sobre o valor original da mensagem oculta. Como mostrado no Algoritmo 1, tanto o vetor contendo o voto do eleitor quanto o somatório dos valores desse vetor acompanham uma prova de conhecimento parcial, mostrando que o voto não viola a definição de voto válido segundo a definição dada em 6.

E como mostrado no Algoritmo 3, o *Counter* realiza o teste de verificação da prova de conhecimento parcial, rejeitando qualquer voto com valores ilegais, mas sem comprometer a privacidade do eleitor. Logo, apenas votos válidos são inclusos na contagem final.

Teorema 4 (Individual Verifiability)**.** *Se o esquema de comprometimento de bits apresenta propriedade de amarração e a máquina de verificação não obtém informação que revele o voto do eleitor, então o eleitor pode verificar a validade do seu voto.*

Pelo Teorema 1, é impossível revelar um valor diferente do comprometido através do esquema de comprometimento de bits incondicional. Além disso a máquina de verificação não possui comunicação com a máquina de eleição. Logo, como o eleitor abre o comprometimento B_i utilizando a máquina de verificação, é possível ter certeza que o valor comprometido é válido.

Teorema 5 (correctness tallying)**.** *Se cada voto é válido, então a exatidão da contagem dos votos pode ser garantida.*

Assuma que todos os votos são válidos. Pela propriedade homomórfica do ElGamal Exponencial, o produto de uma série de cifras decifra para a soma dos valores que foram cifrados. Logo, como os votos são cifrados através do ElGamal Exponencial e se 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 6 (individual verifiability)**.** *Se o voto do eleitor foi aceito pelo Counter, então o eleitor pode verificar que seu voto foi incluído na contagem.*

Assuma que o voto do eleitor foi aceito pelo *Counter*. Como descrito no capítulo 3.3, o eleitor leva consigo um recibo contendo a tupla $< ID_i, H_i, B_i >$, onde o comprometimento de bits de seu voto é representado por B_i e as autoridades publicam uma tabela contendo todas as tuplas $< ID, H, B >$ aceitas pelo *Counter*. Logo, todo eleitor em posse do seu recibo e com acesso à tabela pode conferir a inclusão do seu voto.

Teorema 7 (privacy)**.** *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.*

A prova deriva diretamente do fato de os valores representando os votos dos eleitores divulgados no canal público são gerados a partir do esquema de comprometimento de bits incondicionalmente seguro.

Teorema 8 (Universal Verifiability)**.** *Se as Autoridades ou o Counter alteram o resultado da eleição, então qualquer eleitor pode mostrar que o resultado da eleição foi alterado.*

Assuma que as Autoridades ou o *Counter* alteram o resultado da eleição. Pela propriedade de amarração do esquema de comprometimento de bits incondicional, não é possível revelar um valor diferente do comprometido. Logo, partindo do fato que os valores publicados após a eleição são gerados a partir do esquema de comprometimento de bits incondicional, conclui-se que a corretude do resultado pode ser verificada através da Equação 11.

Capítulo 5

Análise de Desempenho

Nesse capítulo apresentamos uma análise do desempenho dos algoritmos utilizados neste trabalho. Os testes foram realizados com chave pública de tamanhos 512 bits (apresentando segurança provável com validade entre 30 e 50 anos, segundo (SMART et al., 2018)) e 1024 bits. Os valores são mostrados em segundos, com desvio padrão σ 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 baixada no endereço <https://github.com/pdrvncs/UrnaPrototipo>.

5.1 Algoritmo de geração do voto

A Tabela 5.1 apresenta o tempo em segundos para computar o Algoritmo 1 (executado pela urna), para um tamanho de chave k e um número N de candidatos.

Tabela 4 – Desempenho do algoritmo de geração do voto

	N = 100	N = 1000
k = 512	0.82 ($\sigma = 0.02$)	7.87 ($\sigma = 0.21$)
k = 1024	4.77 ($\sigma = 0.03$)	47.44 ($\sigma = 0.48$)

5.2 Algoritmo de processamento do voto

A Tabela 5.2 apresenta o tempo em segundos para computar o Algoritmo 3 (executado pelo *Counter*), para um tamanho de chave k e uma quantidade de candidatos N.

Tabela 5 – Desempenho do algoritmo do *Counter*

	N = 100	N = 1000
k = 512	0.75 ($\sigma = 0.01$)	7.36 ($\sigma = 0.35$)
k = 1024	4.38 ($\sigma = 0.18$)	43.38 ($\sigma = 1.56$)

5.3 Algoritmo de obtenção do resultado

A Tabela 5.3 apresenta o tempo em segundos para computar o Algoritmo 4 de obtenção do resultado das eleições, para um tamanho de chave k e uma quantidade de eleitores I.

Tabela 6 – Desempenho do algoritmo de obtenção dos resultados

	I = 100	I = 1000
k = 512	0.09 ($\sigma = 0.01$)	0.91 ($\sigma = 0.03$)
k = 1024	0.52 ($\sigma = 0.01$)	5.04 ($\sigma = 0.04$)

5.4 Algoritmo de auditação

A Tabela 5.4 apresenta o tempo em segundos para computar o Algoritmo 5, para um tamanho de chave k e uma quantidade de eleitores I.

Tabela 7 – Desempenho do algoritmo de auditação

	I = 100	I = 1000
k = 512	0.01 ($\sigma = 0.01$)	0.68 ($\sigma = 0.29$)
k = 1024	0.33 ($\sigma = 0.01$)	2.63 ($\sigma = 0.03$)

Capítulo 6

Conclusão

Neste trabalho usamos encriptação homomórfica e esquemas de comprometimento para construir uma urna eletrônica segura e transparente. Provou-se que a urna atende aos requisitos do modelo de urna segura proposto pela comunidade científica. Além disso, os resultados dos experimentos mostraram a viabilidade de verificação do resultado da eleição por qualquer uma das partes externas (eleitor, observador internacional).

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 como mostrado, qualquer parte com acesso à internet pode auditar o resultado da eleição. Por outro lado, o uso de esquemas de comprometimento habilitou qualquer eleitor verificar a inclusão e a integridade de qualquer voto, sem comprometer o seu sigilo que é exigido no processo eleitoral.

6.1 Trabalhos Futuros

E apesar dos bons resultados preliminares quanto a segurança da urna, ainda são necessários mais testes, nós não testamos a escalabilidade do esquema de votação e o impacto disso na transparência e na segurança de uma eleição. Futuramente, pretendemos projetar uma urna baseada em métodos de encriptação baseados em reticulados (PINO et al., 2017).

Referências

AZOUGAGHE, A.; HEDABOU, M.; BELKASMI, M. An electronic voting system based on homomorphic encryption and prime numbers. In: **2015 11th International Conference on Information Assurance and Security (IAS)**. [S.l.]: IEEE, 2015. p. 140–145.

DAMGARD, I.; NIELSEN, J. B. Commitment schemes and zero-knowledge protocols (2011).

ELGAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. **IEEE transactions on information theory**, IEEE, v. 31, n. 4, p. 469–472, 1985.

GRAAF, J. v. d. Long-term threats to ballot privacy. **IEEE Security Privacy**, v. 15, n. 3, p. 40–47, 2017. ISSN 1540-7993.

_____. **O mito da urna : desvendando a (in)segurança da urna eletrônica**. 1. ed. [S.l.: s.n.], 2017. 87 p.

LIMA, F. T. et al. Urna eletrônica de terceira geração: Um protótipo para eleicções auditáveis. In: **XVII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais: SBSEG 2017**. Brasília-DF: [s.n.], 2017. p. 677–685.

MATIAS, D. F. A. e Pedro Barbosa e Thiago N. C. Cardoso e Caio Luders e P. Execução de código arbitrário na urna eletrônica brasileira. **Anais do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)**, p. 57–70, 2018. Disponível em: <http://portaldeconteudo.sbc.org.br/index.php/sbseg/article/view/4243>.

PENG, K. et al. Efficient multiplicative homomorphic e-voting. In: **Information Security**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 381–393. ISBN 978-3-642-18178-8.

PINO, R. del et al. Practical quantum-safe voting from lattices. In: **Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security**. New York, NY, USA: ACM, 2017. (CCS '17), p. 1565–1581. ISBN 978-1-4503-4946-8. Disponível em: <http://doi.acm.org/10.1145/3133956.3134101>.

SMART, N. P. et al. **Algorithms, Key Size and Protocols Report (2018)**. [S.l.], 2018.

SMITH, W. D. Cryptography meets voting. **September**, v. 10, p. 80, 2005.

YANG, X. et al. A secure verifiable ranked choice online voting system based on homomorphic encryption. **IEEE Access**, PP, p. 1–1, 03 2018.