

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

João Pedro Cardoso Barbosa

Votação Eletrônica Pós-quântica Aplicada no Helios Voting

Florianópolis
2024

João Pedro Cardoso Barbosa

Votação Eletrônica Pós-quântica Aplicada no Helios Voting

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Prof^ª. Thaís Bardini Idalino, Dr.

Coorientador: Gustavo Zambonin, Me.

Florianópolis

2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Barbosa, João Pedro Cardoso
Votação Eletrônica Pós-quântica Aplicada no Helios Voting /
João Pedro Cardoso Barbosa ; orientadora, Thaís Bardini
Idalino, coorientador, Gustavo Zambonin, 2024.
117 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Sistemas de Informação, Florianópolis, 2024.

Inclui referências.

1. Sistemas de Informação. 2. criptografia. 3. votação
eletrônica. 4. criptografia pós-quântica. 5. homomorfismo.
I. Idalino, Thaís Bardini. II. Zambonin, Gustavo. III.
Universidade Federal de Santa Catarina. Graduação em
Sistemas de Informação. IV. Título.

João Pedro Cardoso Barbosa
Votação Eletrônica Pós-quântica Aplicada no Helios Voting

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo curso de Graduação em Sistemas de Informação.

Florianópolis, 8 de Julho de 2024.

Prof. Álvaro Júnio Pereira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof^a. Thaís Bardini Idalino, Dr.
Orientadora
Universidade Federal de Santa Catarina

Gustavo Zambonin, Me.
Coorientador
Universidade Federal de Santa Catarina

Prof. Diego de Freitas Aranha, Dr.
Avaliador
Aarhus University

Prof. Jean Everson Martina, Dr.
Avaliador
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Com imensa gratidão, agradeço minha orientadora Thaís Bardini Idalino pela ajuda e apoio em todas as etapas desse árduo processo. Não achava que encontraria uma pessoa tão atenciosa quanto você para me orientar. Obrigado por me acompanhar e valorizar meu trabalho, por todas as reuniões, pelo auxílio quando estava preso em algo que sozinho não conseguiria resolver e pela ajuda que se estendeu para o desenvolvimento do meu plano de mestrado. Também gostaria de agradecer meu coorientador Gustavo Zambonin pela ajuda no desenvolvimento do trabalho e, especialmente, por acreditar em mim, me proporcionando a oportunidade de fazer parte do LabSEC. Também sou grato por todos os conselhos, auxílio e oportunidades, tanto como acadêmico quanto como amigo.

Gostaria de estender meus agradecimentos aos membros do LabSEC que se tornaram grandes amigos no decorrer destes diversos semestres. Sou grato por nossas conversas, pela ajuda, por sempre escutarem quando extravasava sobre este trabalho e pelo companheirismo. Em especial Anthon Porath Gretter, Anthony Bernardo Kamers, Eduardo de Moraes, Gustavo de Castro Biage, Larissa Gremelmaier Rosa, Lucas Mayr de Athayde, Marcos Tomaszewski e vários outros da equipe que faço parte ou de outras.

Gostaria também de agradecer aos meus amigos que precedem à universidade e amadureceram junto comigo. A dois grupos de amigos que mesmo após vários anos ainda converso diariamente, ambos surgiram por motivos simples e se tornaram algo muito maior. O primeiro que era um grupo para jogarmos composto por Bernardo Schmidt Teixeira, Guilherme Melchioris, João Cândido Gonzaga Brandão, Mateus Hort Monteiro, Tales Guerra Campos, Thiago Vieira Xavier Santana e Vinicius Lohn da Silva. E, o segundo composto por Mateus Saidel e Pedro Afonso de Castro Sandreschi para combinarmos de jogar basquete quando convém. Para ambos os grupos sou grato por zelarem por mim e me aceitarem. Desejo profundamente que todos continuem fazendo parte da minha jornada. Gostaria em especial de agradecer Augusto Pamplona por me inspirar na escolha do curso de Sistemas de Informação e por fazer parte da minha história.

Com emoção, desejo agradecer minha família, meus avós, tios, primas e, em especial, a minha mãe Lidiani, meu farol e maior fonte de segurança, e meu pai Laércio, que despertou minha paixão por computadores, sendo minha primeira e maior inspiração para moldar a pessoa e profissional que sou hoje. Ambos sempre me apoiaram e acreditaram em mim. Obrigado pelo suporte nos momentos difíceis e estressantes. Tenho orgulho em ser filho de vocês e gratidão pelo amor incondicional que demonstram por mim, pelos sacrifícios que fizeram e por me proporcionarem tantas oportunidades. Também gostaria de agradecer em particular minha avó Arlete, por todo o apoio na minha jornada estudantil e pessoal. Acredito que uma parte da minha personalidade espelho de ti e qualquer conquista acadêmica minha também é uma conquista sua.

"This is my song; And no one can take it away."
Labi Siffre

RESUMO

A digitalização do processo de votação traz comodidades tanto no lado do votante, quanto para as entidades responsáveis pela contagem de votos. No entanto, votação eletrônica introduz um alto grau de complexidade na busca pelo equilíbrio entre seu altíssimo requisito de privacidade com a necessidade da verificabilidade das etapas do processo, demandando propriedades dificilmente alcançáveis em conjunto, como anonimato de votos, cédulas verificáveis, proteção contra fraudes e auditabilidade. Assim, criptografia é um instrumento fundamental para possibilitar que esses protocolos atendam esses requisitos, proporcionando artifícios para cifra de votos juntamente com provas de conhecimento zero que garantem asserções confiáveis referentes à validade da eleição. Muito utilizado no ambiente acadêmico, o sistema Helios desenvolvido por Adida (2008) emprega o protocolo de votação eletrônica de Cramer et al. (1997) que é baseado em contagem homomórfica das cifras do esquema ElGamal, permitindo a computação dos votos sem o resultado de um voto individual ser revelado, garantindo a privacidade do votante. Todavia, o advento da computação quântica compromete a segurança dos algoritmos clássicos como o ElGamal, demandando esforços para padronização de novos esquemas baseados em problemas seguros contra este tipo de adversário, notavelmente as classes de problemas baseadas em reticulados. Dessa forma, este trabalho realizará um estudo de protocolos clássicos de votação eletrônica e das propostas na literatura de esquemas seguros contra adversários quânticos, selecionando um para implementação, considerando o Helios como caso de uso.

Palavras-chave: Votação Eletrônica. Homomorfismo. Criptografia Pós-Quântica. Criptografia Baseada em Reticulados.

ABSTRACT

The digitalization of the voting process brings convenience both for the voter and the entities responsible for counting the votes. However, electronic voting introduces a high degree of complexity in the search for a balance between its extremely high privacy requirement and the need for verifiability in all the process stages. It demands properties that are difficult to achieve together, such as anonymity of votes, verifiable ballots, protection against fraud, and auditability. Thus, cryptography is a fundamental instrument that enables these protocols to meet these requirements, providing mechanisms for encrypting votes along with zero-knowledge proofs that are reliable assertions regarding the validity of the election. Widely used in academia, the Helios system developed by Adida (2008) employs the electronic voting protocol of Cramer et al. (1997), which is based on homomorphic counting of ElGamal's ciphers, allowing votes to be computed without the result of an individual vote being revealed, guaranteeing voter privacy. However, the advent of quantum computing compromises the security of classical algorithms such as ElGamal, requiring efforts to standardize new schemes based on secure problems against this type of adversary, notably classes of problems based on lattices. Therefore, this work will study classic electronic voting protocols and proposals in the literature for secure schemes against quantum adversaries, selecting one for implementation, considering Helios as a use case.

Keywords: Electronic Voting. Homomorphism. Post-Quantum Cryptography. Lattice-Based Cryptography.

LISTA DE FIGURAS

Figura 1 – Sistema multi-autoridade	26
Figura 2 – <i>Shortest Vector Problem</i> (SVP) para um reticulado de duas dimensões	41
Figura 3 – <i>Closest Vector Problem</i> (CVP) para um reticulado de duas dimensões	42
Figura 4 – Exemplo de aplicação para <i>Homomorphic Encryption</i> (HE)	44
Figura 5 – Protocolo- Σ de conhecimento-zero para $\log_g y_j = \log_x w_j$	50
Figura 6 – Protocolo multi-autoridade homomórfico de Cramer	52
Figura 7 – Protocolo- Σ de conhecimento-zero para encriptação da cédula	54
Figura 8 – Protocolo Π_{Lin}	66
Figura 9 – Protocolo Π_{Sum} , alterações realizadas pelo autor em relação ao Π_{Lin} estão marcadas em vermelho	67
Figura 10 – Protocolo $\Pi_{Shuffle}$	67
Figura 11 – Fase de configuração	73
Figura 12 – Fase de registro	74
Figura 13 – Fase de votação	75
Figura 14 – Fase de contagem	75
Figura 15 – Interface para cadastro de uma nova eleição	78
Figura 16 – Interface para cadastro de questões	79
Figura 17 – Estrutura de uma questão	80
Figura 18 – Interface para registro de votantes	80
Figura 19 – Interface para configuração de autoridades	81
Figura 20 – Cabine eleitoral	81
Figura 21 – Interface para responder uma questão	82
Figura 22 – Interface para revisão das repostas	82
Figura 23 – Interface para depósito da cédula	83
Figura 24 – Estrutura da cédula encriptada de uma questão	83
Figura 25 – Exemplo de contagem	84
Figura 26 – Mapeamento de estrutura do esquema de comprometimento para uma classe	85
Figura 27 – Código para o algoritmo Setup	89
Figura 28 – Código para o algoritmo Register	89
Figura 29 – Código para cifragem de aberturas dos comprometimentos	90
Figura 30 – Excerto da classe que representa um participante do protocolo	90
Figura 31 – Implementação da fase de configuração	91
Figura 32 – Excerto da implementação da fase de votação	91
Figura 33 – Novas tabelas para o protocolo pós-quântico	93
Figura 34 – Serialização de uma classe	94

LISTA DE TABELAS

Tabela 1 – Impacto de computadores quânticos nos algoritmos criptográficos atuais . . .	39
Tabela 2 – Visão geral dos esquemas de votação eletrônica pós-quânticos	60
Tabela 3 – Comparação de performance dos esquemas	61
Tabela 4 – Parâmetros dos esquemas de comprometimento e encriptação	64
Tabela 5 – Valores dos parâmetros dos esquemas de comprometimento e encriptação .	86
Tabela 6 – Desempenho dos algoritmos	92
Tabela 7 – Comparação do desempenho entre protocolos	92

ACRÔNIMOS

SVP_γ *Approximate Shortest Vector Problem*

CVP *Closest Vector Problem*

E2E-V *End-to-End Verifiable*

EVOLVE *Electronic Voting from Lattices with Verification*

FHE *Fully Homomorphic Encryption*

HE *Homomorphic Encryption*

IACR *International Association for Cryptologic Research*

IBE *Identity Based Scheme*

LWE *Learning With Erros*

NIST *National Institute of Standards and Technology*

PHE *Partially Homomorphic Encryption*

SIS *Short Integer Solution*

SVP *Shortest Vector Problem*

SWHE *Somewhat Homomorphic Encryption*

SUMÁRIO

1	INTRODUÇÃO	25
1.1	OBJETIVOS	26
1.1.1	Objetivos Gerais	26
1.1.2	Objetivos Específicos	27
2	FUNDAMENTAÇÃO TEÓRICA	29
2.1	ESTRUTURAS ALGÉBRICAS	29
2.1.1	Notações	32
2.2	CRIPTOGRAFIA	32
2.2.1	Família de Funções	33
2.2.2	Criptografia de Chave Pública	34
2.2.3	Prova de conhecimento zero	36
2.2.3.1	<i>Prova de conhecimento zero não interativa</i>	37
2.2.4	Esquema de comprometimento	38
2.3	CRIPTOGRAFIA PÓS-QUÂNTICA	38
2.3.1	Reticulados	40
2.3.1.1	<i>Problemas Computacionais</i>	41
2.3.1.2	<i>Short Integer Solution (SIS)</i>	42
2.3.1.3	<i>Learning With Errors (LWE)</i>	42
2.3.1.4	<i>Ring-SIS</i>	43
2.3.1.5	<i>Ring-LWE</i>	43
2.4	CRIPTOGRAFIA HOMOMÓRFICA	44
2.4.1	Homomorfismo de ElGamal	45
2.5	VOTAÇÃO ELETRÔNICA	46
2.5.1	Propriedades fundamentais	47
2.5.2	Esquema multi-autoridade de Cramer et al. (1997)	49
2.5.2.1	<i>Provas de validade</i>	52
2.5.2.2	<i>Eleições de múltipla escolha</i>	53
2.5.3	Helios	53
2.5.4	Outros protocolos	55
3	TRABALHOS RELACIONADOS	57
3.1	AN HOMOMORPHIC LWE BASED E-VOTING SCHEME	57
3.2	PRACTICAL QUANTUM-SAFE VOTING FROM LATTICES	58
3.3	EPOQUE: PRACTICAL END-TO-END VERIFIABLE POST-QUANTUM- SECURE E-VOTING	58
3.4	LATTICE-BASED PROOF OF SHUFFLE AND APPLICATIONS TO ELEC- TRONIC VOTING	59

3.5	VISÃO GERAL	60
4	PROCOLO SELECIONADO	63
4.1	PRELIMINARES	63
4.2	PARÂMETROS	63
4.3	PRIMITIVAS	64
4.3.1	Esquema de Comprometimento	64
4.3.2	Provas de conhecimento	66
4.3.2.1	<i>Relações lineares</i>	66
4.3.2.2	<i>Soma</i>	66
4.3.2.3	<i>Embaralhamento de valores conhecidos</i>	67
4.3.3	Esquema de Encriptação Verificável	68
4.3.3.1	<i>Cifrando aberturas de comprometimentos</i>	70
4.3.4	Códigos de Retorno	70
4.4	ALGORITMOS	71
4.5	PROCOLO	73
4.5.1	Fase de Configuração	73
4.5.2	Fase de Registro	74
4.5.3	Fase de Votação	74
4.5.4	Fase de Contagem	74
5	HELIOS	77
5.1	AUTENTICAÇÃO	77
5.2	CRIAÇÃO DE UMA ELEIÇÃO	78
5.2.1	Questões	79
5.2.2	Registro de votantes	79
5.2.3	Configuração de autoridades	80
5.3	VOTAÇÃO	81
5.3.1	Verificação de um voto	83
5.4	CONTAGEM	84
6	IMPLEMENTAÇÃO	85
6.1	BIBLIOTECA COMPARTILHADA	85
6.2	ORGANIZAÇÃO DO PACOTE	86
6.2.1	Geração de números aleatórios	87
6.2.2	Estrutura de um voto	88
6.2.3	Função pseudo-aleatória	88
6.2.4	Algoritmos	88
6.2.5	Implementação de teste	89
6.2.6	Testes de desempenho	91
6.2.7	Metodologia	91

6.2.7.1	<i>Resultados</i>	91
6.3	INTEGRAÇÃO COM O SISTEMA HELIOS	93
7	CONCLUSÃO	97
7.1	TRABALHOS FUTUROS	98
	REFERÊNCIAS	99
8	APÊNDICE 1: ARTIGO DO TCC	105
9	APÊNDICE 2: CÓDIGO FONTE	117

1 INTRODUÇÃO

O advento de computadores proporcionou grandes benefícios e comodidades para a população, a maior escalabilidade, confiabilidade e presença de sistemas na sociedade possibilita a realização das mais diversas tarefas de maneira remota do conforto de casa. Assim, é um interesse lógico também digitalizar o processo tradicional de votação que demanda tempo e esforço considerável de todas as partes envolvidas. Todavia, votação eletrônica é um tema extremamente delicado, sendo sucessível a ataques efetivos e de difícil detecção que em outros sistemas (*e.g. e-commerce e e-banking*) são mitigáveis e toleráveis. Integralmente, votação eletrônica é um processo que envolve preocupações humanas, digitais e éticas, não existindo respostas definitivas para todos os problemas presentes em seu meio.

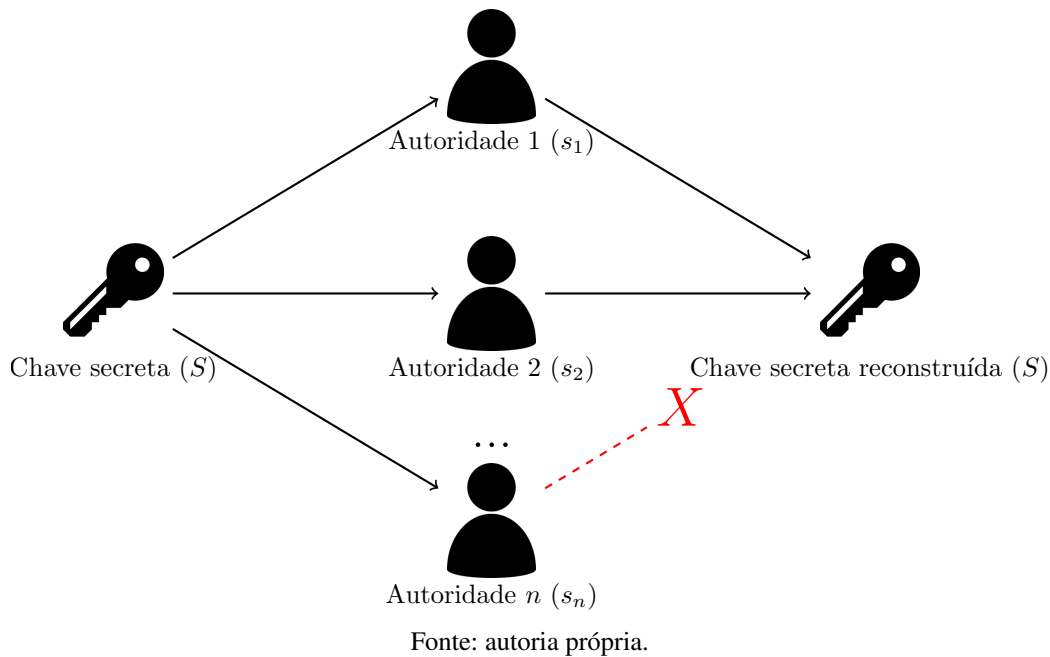
Essencialmente, um sistema de votação deve balancear conveniência, privacidade e verificabilidade, sendo ineficiente se qualquer uma dessas propriedades não estiver em um grau de maturidade aceitável. Esse objetivo é de extrema dificuldade, especialmente considerando que o aperfeiçoamento de uma propriedade pode antagonizar outra. Por exemplo, o aumento de privacidade pode acarretar em uma menor disponibilidade de dados para garantir verificabilidade e um processo de votação mais árduo e inconveniente.

Assim, criptografia é um artifício crucial para desenvolvimento de sistemas de votação eletrônica que adéquam-se a essas propriedades. Uma das principais ferramentas disponíveis é o homomorfismo dos esquemas criptográficos que, caracterizando sucintamente sua importância, permite a contagem de votos sem revelar o candidato de preferência do votante. Tecnicamente, homomorfismo permite a execução de operações sobre textos cifrados que se traduzem para o texto plano.

Helios (ADIDA, 2008) é um dos esquemas de votação eletrônica disponíveis abertamente para o público. Notavelmente, ele tem boas garantias tanto de privacidade quanto verificabilidade em combinação com a conveniência concedida pelo votação remota via sistema web. Todavia, coerção é um aspecto que a aplicação não tenta solucionar (ADIDA, 2008), sendo não obstante para aplicações onde o risco desse acontecimento é baixo. Dessa forma, Helios é amplamente utilizado, especialmente no meio acadêmico em (ADIDA et al., 2009) e, neste trabalho, é de interesse enfatizar o uso do sistema pela Universidade Federal de Santa Catarina.

Em seu núcleo, Helios baseia-se no protocolo de Cramer et al. (1997), construído encima do criptossistema de Pedersen (1991) para *secret sharing* e do esquema criptográfico homomórfico ElGamal (ELGAMAL, 1985). As características dessas primitivas permitem que, além dos votos serem computados em sua forma encriptada por meio do homomorfismo, seja necessário um acordo entre uma parcela das partes (autoridades) responsáveis pela contagem. Essa garantia é importantíssima para descentralização da responsabilidade de revelar e auditar o resultado da eleição, impedindo que uma entidade única seja capaz de decifrar os votos individualmente, quebrando a privacidade dos votantes. É possível visualizar o fluxo pela Figura 1, nela temos uma chave secreta S responsável por revelar a contagem de votos que foi dividida entre n autoridades. Mesmo que algumas entidades sejam maliciosas e não desejem contribuir

Figura 1 – Sistema multi-autoridade



para revelação, por exemplo a Autoridade n , ainda é possível obter o resultado desejado.

Atualmente, as garantias criptográficas permitem ao Helios operar com confiabilidade, mantendo a privacidade do votante e a integridade do resultado final. No entanto, a segurança do sistema é construída com base em premissas clássicas que podem ser comprometidas em tempo polinomial por um computador quântico suficientemente poderoso aplicando o algoritmo de Shor (SHOR, 1999). Assim, existe um esforço por parte do *National Institute of Standards and Technology* (NIST) para o desenvolvimento de esquemas baseados em conjecturas resistentes ao advento da computação quântica.

Entre as famílias de algoritmos com a propriedade de ser resistente a algoritmos quânticos, temos os reticulados. Que, além de serem baseados em conjecturas fortes, permitem o desenvolvimento de instrumentos desejáveis para a construção de esquemas de votação eletrônica. Permitindo que protocolos realizem a contagem homomórfica, semelhante ao Helios, ou utilizem outros artifícios que garantem que um voto seja privado e corretamente calculado. Assim, é desejável a adequação do protocolo do Helios com primitivas criptográficas baseadas em reticulados como ação preemptiva a ataques de computadores quânticos.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Realizar um estudo teórico de votação eletrônica e como adequar as propriedades pre-estabelecidas na literatura ao advento da computação quântica. Analisando como caso de uso o sistema Helios e substituindo seu protocolo criptográfico existente por um construído acima

de esquemas votação eletrônica baseados em reticulados.

1.1.2 Objetivos Específicos

- Expor os conceitos clássicos de votação eletrônica, demonstrando as principais primitivas criptográficas que tornam o processo confiável;
- Explicar propostas de protocolos pós-quânticos de votação eletrônica baseados em reticulados presentes na literatura;
- Selecionar um dos protocolos para realização de uma análise aprofundada e implementação, expondo as primitivas criptográficas que permitem a formação de um sistema multi-autoridade e a realização de operações em texto encriptado;
- Aplicação do protocolo selecionado no sistema do Helios.
- Avaliação de métricas mensuráveis do esquema como desempenho e tamanho dos votos;

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ESTRUTURAS ALGÉBRICAS

Esta seção apresenta as estruturas algébricas necessárias para fundamentação das primitivas criptográficas que serão abordadas subsequentemente, sendo fundamentalmente baseado nos livros de Hoffman (1971) e Roman et al. (2005) e nas notas de aula de Forney (2003). Destacam-se Grupos Cíclicos (Definição 2.1.5) para instanciação do esquema ElGamal, espaços euclidianos (Definição 2.1.8) para contextualização de reticulados e anéis (Definição 2.1.11) que são utilizados em variações de problemas computacionais relevantes em reticulados. Adicionalmente, serão expostas outras estruturas necessárias para auxiliar a definição dos conceitos abordados acima.

Definição 2.1.1 (Grupo). Um grupo $(G, *)$ é uma dupla contendo os seguintes elementos:

1. Um conjunto G .
2. Uma operação $*$ que associa cada par $x, y \in G$ a um elemento $xy = x * y \in G$ de forma que as seguintes propriedades sejam estabelecidas:
 - a) **Associatividade:** $x * (y * z) = (x * y) * z$ para todo $x, y, z \in G$.
 - b) **Elemento Neutro:** existe um elemento único $e \in G$ de forma que $e * x = x * e = x$ para todo $x \in G$.
 - c) **Elemento Inverso:** Para cada $x \in G$, existe um elemento único $x^{-1} \in G$ tal que $x * x^{-1} = x^{-1} * x = e$.

Definição 2.1.2 (Grupo Abelian). um grupo $(G, *)$ é abeliano se sua operação possui a propriedade adicional de comutatividade, de forma que $x * y = y * x$ para todo $x, y \in G$.

Definição 2.1.3 (Subgrupo). Dado grupo $(G, *)$, a tupla $(H, *)$ é um subgrupo de G se:

- $H \subseteq G$.
- H também é um grupo para a operação $*$.

Definição 2.1.4 (Conjunto gerador). Um conjunto gerador X para um grupo $(G, *)$, é um subconjunto de G de forma que qualquer elemento do grupo pode ser expressado como a combinação de um número finito de elementos de X e seus inversos sob a operação $*$. Assim, pode-se dizer que X gerou G , matematicamente definido como $G = \langle X \rangle$.

Definição 2.1.5 (Grupo cíclico). Um grupo $(G, *)$ é considerado cíclico se ele foi gerado por apenas um elemento x , denominado de **elemento gerador**, ou seja seu conjunto gerador é $X = \{x\}$, podemos definir essa relação matematicamente como $G = \langle x \rangle$.

Definição 2.1.6 (Corpo). Um corpo $(\mathbb{F}, \oplus, \otimes)$ é um conjunto \mathbb{F} com pelo menos dois elementos e duas operações \oplus e \otimes , de forma que as seguintes propriedades sejam estabelecidas:

1. O conjunto \mathbb{F} forma um grupo abeliano sob a operação \oplus e com o elemento neutro e .
2. O conjunto $\mathbb{F} \setminus \{0\}$ forma um grupo abeliano sob a operação \otimes e com elemento neutro o .
3. **Lei distributiva:** para todo $a, b, c \in \mathbb{F}$, $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$.

Definição 2.1.7 (Espaço Vetorial). Seja $F = (\mathbb{F}, \oplus, \otimes)$ um corpo, um espaço vetorial $(\mathbb{V}, +, \times, F)$ sobre F é um conjunto de vetores \mathbb{V} junto de duas operações.

1. A primeira $+$ chama-se de adição vetorial. Devendo respeitar a seguinte propriedade:
 - a) $(\mathbb{V}, +)$ deve ser um grupo abeliano.
2. A segunda \times chama-se multiplicação escalar. Para todo $a, b \in \mathbb{F}$ e todo $u, v \in \mathbb{V}$, devem ser respeitadas as seguintes propriedades:
 - a) **Compatibilidade de multiplicação escalar com multiplicação no corpo:**
 $a \times (b \times v) = (a \otimes b) \times v$.
 - b) **Distributividade da multiplicação escalar em relação à adição vetorial:**
 $a \otimes (u + v) = a \otimes u + a \otimes v$.
 - c) **Distributividade de multiplicação escalar em relação à adição no corpo:**
 $(a \oplus b) \times v = a \times v + b \times v$.
 - d) **Elemento neutro da multiplicação escalar:** $ov = v$, onde o denota o elemento neutro da operação \otimes sobre o corpo F .

Definição 2.1.8 (Espaço Euclidiano). Para um $n \in \mathbb{Z}$ e um corpo $F = (\mathbb{R}, +, \times)$, um espaço euclidiano de dimensão n é um espaço vetorial $(\mathbb{R}^n, \oplus, \otimes, F)$ que, para todos os vetores $v, u \in \mathbb{R}^n$ e para todo $a \in \mathbb{R}$, as seguintes propriedades são respeitadas sob as operações:

1. **Adição vetorial:** $v \oplus u = (v_1 + u_1, v_2 + u_2, \dots, v_n + u_n)$.
2. **Multiplicação escalar:** $a \otimes v = (a \times v_1, a \times v_2, \dots, a \times v_n)$.

Adicionalmente, é definida a operação de produto interno $\langle v, u \rangle$ denotada como:

$$\langle v, u \rangle = (v_1 \times u_1 + v_2 \times u_2 + \dots + v_n \times u_n).$$

Definição 2.1.9 (Norma- p). Para $p \geq 1$ pertencente aos números reais, e um espaço vetorial de n dimensões \mathbb{R}^n , a norma- p de um vetor $v \in \mathbb{R}^n$ é dada pela fórmula:

$$\|v\|_p = \left(\sum_{i=1}^n |v_i|^p \right)^{1/p}.$$

Com p aproximando-se do infinito, o resultado da equação aproxima-se do valor absoluto do elemento máximo do vetor. Assim, existe o caso especial para norma- ∞ , onde:

$$\|v\|_{\infty} = \max_{i \in \{1, \dots, n\}} |x_i|.$$

Definição 2.1.10 (Norma Euclideana). Em um espaço euclidiano de n dimensões \mathbb{R}^n , a noção intuitiva de comprimento de qualquer vetor $v \in \mathbb{R}^n$ é dada pela norma euclideana, também conhecida como norma-2, denotada como $\|v\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$.

Definição 2.1.11 (Anel). Um anel $(R, +, \times)$ é uma tripla contendo um conjunto R equipado com dois operadores binários, o de adição $+$ e o de multiplicação \times , de forma que os três axiomas a seguir sejam satisfeitos:

1. **Grupo abeliano:** R é um grupo abeliano para a operação de adição $+$, ou seja, para todo elemento em R temos as propriedades de associatividade, comutatividade, elemento neutro e elemento identidade;
2. **Associatividade da multiplicação:** a operação de \times é associativa para R , ou seja, para todo $x, y, z \in R$, $(x \times y) \times z = x \times (y \times z)$ e existe um elemento neutro $e \in R$ de forma que $x \times e = e \times x = x$;
3. **Distributividade da multiplicação em respeito a adição:** a operação \times é distributiva com respeito a $+$, ou seja, para todo $x, y, z \in R$, $x \times (y + z) = (x \times y) + (x \times z)$ e $(y + z) \times x = (y \times x) + (z \times x)$.

Definição 2.1.12 (Anel polinomial). Para um anel $(R, +, \times)$, seu anel de polinômios $R[x]$ é denotado como todos polinômios de indeterminada x com coeficientes pertencentes a R . Assim, para toda sequência $a_0, a_1, \dots, a_{\infty} \in R$, os polinômios $f(x)$ de $R[x]$ são definidos como

$$f(x) = a_0x^0 + a_1x^1 + a_2x^2 + \dots + a_{\infty}x^{\infty}.$$

Definição 2.1.13 (Produto Tensorial (ARANHA et al., 2021)). Sejam $\alpha, \beta, \gamma, \delta \in \mathbb{Z}^+$ e S um anel arbitrário. Para duas matrizes $\mathbf{A} \in S^{\alpha \times \beta}$, $\mathbf{B} \in S^{\gamma \times \delta}$, o produto tensorial denotado como $\mathbf{A} \otimes \mathbf{B} \in S^{(\alpha \cdot \gamma) \times (\beta \cdot \delta)}$ é dado por:

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} b_{1,1} \cdot \mathbf{A} & \dots & b_{1,\delta} \cdot \mathbf{A} \\ \vdots & \ddots & \vdots \\ b_{\gamma,1} \cdot \mathbf{A} & \dots & b_{\gamma,\delta} \cdot \mathbf{A} \end{pmatrix}.$$

Definição 2.1.14 (Função gaussiana (WEISSSTEIN, 2012)). Uma gaussiana é uma função $f(x)$ de formato

$$f(x) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right),$$

de tal forma que $a, b, c \in \mathbb{R}$ e $c > 0$. A função é geralmente utilizada para representar uma distribuição normal, neste caso, temos $c = \sigma$ representando a variância e $b = \mu$ o valor esperado. Assim, temos uma gaussiana de forma

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Existem situações que demandam uma variação da gaussiana com valores discretos. Neste caso, considerando $R \subseteq \mathbb{R}$ como o conjunto de todos os valores possíveis da gaussiana e a função f apresentada acima, uma gaussiana discreta $\mathcal{N}(x)$ é obtida pela normalização de todos os valores de f em R , de tal modo que

$$\mathcal{N}(x) = \frac{f(x)}{f(R)}, \text{ onde } x \in R \text{ e } f(R) = \sum_{x \in R} f(x).$$

2.1.1 Notações

Baseando-se nas notações de Aranha et al. (2021) para amostragem de valores, neste trabalho, se \mathcal{X} é uma distribuição probabilística, então $z \stackrel{\$}{\leftarrow} \mathcal{X}$ representa que z foi amostrado de acordo com a distribuição e, se S é um conjunto finito, $z \stackrel{\$}{\leftarrow} S$ denota que z foi amostrado de S de forma uniformemente aleatória.

2.2 CRIPTOGRAFIA

A criptografia é um resultado da interseção da matemática e da ciência da computação para proteção de informações, transações e comunicações digitais. Nesta seção, serão introduzidos informalmente os principais conceitos e primitivas criptográficas para a fundamentação do trabalho, as definições apresentadas são majoritariamente baseadas no trabalho de Katz e Lindell (2020) e Goldrich (2006, 2009). Primordialmente, quatro pilares formam a base da criptografia moderna que são:

- **Confidencialidade:** Manter o conteúdo da informação disponível apenas para aqueles que tenham autorização para acessá-lo.
- **Integridade:** Garantir que dados não foram alterados de maneira não autorizada e não detectada.
- **Autenticidade:** Identificar e autenticar a origem que gerou os dados.
- **Não-repúdio:** Impedir que uma entidade negue falsamente ter realizado determinada ação.

Esses pilares não são necessariamente cobijáveis ou alcançáveis por todas as primitivas criptográficas, de forma que seja necessário um conjunto delas para formação de um protocolo que cumpra com os requisitos de segurança desejáveis. Por exemplo, funções de resumo criptográfico que serão posteriormente abordadas são capazes de garantir apenas a integridade.

Definição 2.2.1 (Parâmetro de segurança λ). Na abordagem assintótica, podemos definir um parâmetro de segurança λ como o regulador do tempo de execução de todos os algoritmos (KATZ; LINDELL, 2020). Na inicialização de um esquema, o valor é decidido e assume-se que todas as partes, incluindo o atacante, tenham conhecimento sobre ele. Tipicamente, tem-se como requisito o tempo de execução ser polinomial $\lambda^{O(1)}$, ou seja, para alguma constante $c \in \{0, 1, 2, \dots\}$, a complexidade do algoritmo é $O(\lambda^c)$.

Definição 2.2.2 (Oráculo randômico). Um oráculo randômico é uma caixa preta que responde a cada entrada única x com uma saída aleatória y , de forma que o funcionamento interno desta caixa é desconhecido e inescrutável (KATZ; LINDELL, 2020). Consultas ao oráculo são consideradas privadas e caso a entrada se repita, a saída é a mesma.

2.2.1 Família de Funções

Uma família de funções é uma coleção de funções $\mathcal{F} = \{f_k : X \rightarrow Y\}_{k \in K}$ para um espaço de chaves K , domínio X e intervalo Y . Em geral, todas as famílias são implicitamente indexadas por λ . Destacam-se funções de resumo criptográfico (Definição 2.2.6) e funções pseudo-aleatórias (Definição 2.2.7) para o funcionamento interno dos protocolos de votação.

Definição 2.2.3. Uma função $f_k : X \rightarrow Y$ é de caminho único se dado $k \in K$ e $y \in Y$, é inviável descobrir qualquer pré-imagem $x \in X$ de forma que $f^{-1}(y) = x$ (GOLDREICH, 2006), em outras palavras, a função é difícil de inverter. Essa propriedade também chama-se de resistência à pré-imagem.

Definição 2.2.4. Uma função $f_k : X \rightarrow Y$ é resistente à colisão caso seja inviável encontrar $x_1, x_2 \in X$ tal que $f(x_1) = f(x_2)$ (KATZ; LINDELL, 2020).

Definição 2.2.5. Uma função alçapão f é um caso especial da Definição 2.2.3 de forma que exista uma informação secreta s que torne $f_s^{-1}(y) = x$ calculável em tempo polinomial (SCHNEIER, 1996).

Definição 2.2.6. Uma função $f_k : X \rightarrow Y$ é chamada de função de resumo criptográfico caso ela comprima a entrada $|X| > |Y|$ e atendas as seguintes propriedades de segurança (KATZ; LINDELL, 2020):

- **Resistência à colisão:** Definição 2.2.4.
- **Resistência à pré-imagem:** Definição 2.2.3.

A chave $k \in K$ que indexa a função de resumo criptográfico é fundamentalmente diferente do conceito tradicional de uma chave, sendo utilizada apenas para especificar uma função f_k em sua família \mathcal{F} .

Definição 2.2.7. Uma família de funções pseudo-aleatórias PRF é uma coleção de funções que emulam o comportamento de um oráculo randômico (Definição 2.2.2) (GOLDREICH et al., 1986). De forma que, a partir de uma chave k e uma entrada x , a função $\text{PRF}_k(x) \rightsquigarrow y$ é computacionalmente indistinguível de uma saída verdadeiramente aleatória (KELSEY et al., 2016).

2.2.2 Criptografia de Chave Pública

Criptografia de chaves públicas utiliza duas chaves diferentes, uma pública e outra privada, sendo computacionalmente difícil deduzir a chave privada a partir da pública. O algoritmo fundamental do sistema é o gerador de chaves dado o parâmetro de segurança $K(\lambda)$ com o valor das demais variáveis para a geração de chaves sendo definido a partir de λ . As duas definições seguintes baseiam-se em (GOLDREICH, 2006).

Definição 2.2.8 (Sistema de Encriptação de Chave Pública). Um sistema de encriptação de chave pública é uma tripla (K, E, D) de algoritmos:

- Geração de chaves $K(\lambda)$: recebe o parâmetro de segurança λ e retorna (sk, pk) que correspondem a chave privada e a chave pública respectivamente.
- Cifragem $E(pk, M)$: o algoritmo recebe a chave pública pk e uma mensagem M e retorna o texto cifrado C .
- Decifragem $D(sk, C)$: o algoritmo recebe a chave privada sk e um texto cifrado C e retorna a mensagem M .

Essencialmente, o processo utiliza o par de chaves para cifrar e decifrar mensagens sendo normalmente baseado em funções alçapão (Definição 2.2.5), encriptação é o caminho fácil que demanda a chave pública, abertamente distribuída entre todas as partes. A decifração é a direção difícil, com o segredo para solucioná-la sendo a chave privada.

Definição 2.2.9 (Sistema de Assinatura Digital). Um sistema de assinatura digital é uma tripla (K, S, V) de algoritmos:

- Geração de chaves $K(\lambda)$: recebe o parâmetro de segurança λ e retorna (sk, pk) que correspondem a chave privada e a chave pública respectivamente.
- Assinar $S(sk, M)$: o algoritmo recebe a chave privada sk e uma mensagem M e retorna a assinatura do conteúdo σ .
- Verificar $V(pk, M, \sigma)$: o algoritmo recebe a chave pública pk , a assinatura σ e a mensagem M e retorna se a assinatura é válida para dada mensagem e chave.

O processo utiliza o par de chaves para assinar e verificar mensagens, sendo que, decorrente da chave privada ser de conhecimento apenas do assinante, é possível autenticar sua

identidade no momento da verificação. Adicionalmente, garante-se o não-repúdio de forma que o assinante não pode negar a autoria da assinatura e a integridade já que alterações na mensagem M produzirão em $V(pk, M, \sigma)$ um retorno indicando a invalidade da assinatura.

Apresentado por Rivest et al. (1978b), o esquema RSA foi um dos primeiros protocolos de chaves públicas criados, este, baseado na dificuldade do problema de fatoração de números inteiros, é de grande importância tanto como esquema de encriptação, quanto de assinatura digital. Outro esquema de maior importância para o presente trabalho é o ElGamal (ELGAMAL, 1985), firmado no problema do logaritmo discreto sobre um grupo cíclico, seu protocolo completo será apresentado em seguida.

Protocolo 1. Esquema ElGamal

1. Geração de chaves $K(\lambda)$.

Entrada: λ .

Saída: (sk, pk)

- 1) escolha um grupo cíclico $G = \langle g \rangle$ de ordem p e elemento gerador g .
- 2) seja x um número selecionado aleatoriamente em \mathbb{Z}_p , a chave privada $sk = x$.
- 3) calcule a chave pública como $pk = y = g^x \pmod p$.

2. Cifragem $E(pk, M)$.

Entrada: chave pública $pk = y$, mensagem $M \in G$.

Saída: texto cifrado (a, b) .

- 1) selecione um valor aleatório $k \in \mathbb{Z}$.
- 2) compute a, b :

$$\begin{aligned} a &= g^k \pmod p \\ b &= y^k M \pmod p. \end{aligned}$$

3. Decifragem $D(sk, C)$.

Entrada: chave secreta $sk = x$, texto cifrado $C = (a, b)$.

Saída: mensagem $M \in G$.

- 1) compute a mensagem como $M = \frac{b}{a^x} \pmod p$.

Demonstração.

$$\begin{aligned}
 D(C) &= \frac{b}{a^x} \pmod{p} \\
 &= \frac{y^k M}{(g^k)^x} \pmod{p} \\
 &= \frac{(g^x)^k M}{(g^k)^x} \pmod{p} \\
 &= \frac{g^{kx} M}{g^{kx}} \pmod{p} \\
 &= M \pmod{p}.
 \end{aligned}$$

□

Uma das principais conclusões que pode-se retirar da apresentação deste esquema é que ele é probabilístico, de forma que uma mensagem encriptada sob uma mesma chave pode produzir múltiplos texto cifrados distintos, decorrentes da seleção aleatória do parâmetro k na encriptação. Esta propriedade é muito útil para sistemas de votação eletrônica, considerando que é extremamente comum a repetição de votos, que poderia revelar informações indesejadas caso ocorra colisão entre os textos cifrados dos votos.

2.2.3 Prova de conhecimento zero

Provas de conhecimento zero foram inicialmente introduzidas por Goldwasser et al. (1989) e, desde então, são valiosas ferramentas criptográficas que provêm evidências convincentes de que determinada asserção é verdadeira, de forma que o protocolo é equivalente a uma parte confiável garantindo a validade da asserção (GOLDREICH, 2006). Esses sistemas podem ser definidos como a comunicação entre duas partes que possuem conhecimento sobre certas informações públicas: o provador \mathcal{P} que deseja revelar a validade de alguma afirmação pre-determinada ao verificador \mathcal{V} sem produzir nenhum outro conhecimento além da corretude da proposição.

Provas de conhecimento zero são sistemas interativos, ou seja, ocorre uma troca de movimentos entre \mathcal{P} e \mathcal{V} , na qual, aleatoriedade normalmente está presente, tanto para ocultar os valores secretos, quanto para manter o comportamento honesto. Quando \mathcal{V} publica seus valores aleatórios, o protocolo é classificado como *public-coin*, caso contrário, *private-coin*.

Duas principais propriedades devem ser garantidas em provas de conhecimento zero (GOLDREICH; OREN, 1994), 1) *completeness*, se a proposição é verdadeira, um \mathcal{V} honesto será convencido por um \mathcal{P} honesto da validade da asserção; e 2) *soundness*, se a proposição é falsa, é computacionalmente inviável um \mathcal{P} desonesto convencer um \mathcal{V} honesto que ela é verdadeira.

Definição 2.2.10 (Protocolo- Σ). Um protocolo- Σ é uma estrutura comum para sistemas de prova interativas sendo uma especialização de protocolos *public-coin* que envolve três envios de mensagem, consistindo de um comprometimento *com* de \mathcal{P} , um desafio *ch* de \mathcal{V} e uma respostas *resp* de \mathcal{P} .

Exemplo 2.2.1 (Prova interativa de um logaritmo discreto). Utilizada no algoritmo de assinatura de Schnorr (HAO, 2017), esta prova contempla como valores públicos um número primo q , um gerador g de um grupo multiplicativo $\mathbb{Z}_q^* = \{1, \dots, q-1\}$ e $y \in \mathbb{Z}_q^*$ e, como valor privado, o logaritmo discreto $\log_g y$. Considerando um \mathcal{P} que deseja provar que ele tenha conhecimento de um valor x que satisfaça $y = g^x \pmod q$, o seguinte protocolo- Σ é aplicado:

1. \mathcal{P} amostra aleatoriamente $v \xleftarrow{\$} \mathbb{Z}_q^*$ e computa seu comprometimento $com = g^v \pmod q$ que é enviado para \mathcal{V} .
2. \mathcal{V} seleciona um desafio aleatório $ch \xleftarrow{\$} \mathbb{Z}_q^*$ e envia-o para \mathcal{P} .
3. \mathcal{P} computa sua resposta $resp = v - ch \cdot x \pmod q$ e envia-a para \mathcal{V} .
4. \mathcal{V} aceita se $com \stackrel{?}{=} g^{resp} \cdot y^{ch} \pmod q$.

Demonstração. Considerando os valores dos elementos da prova: $y = g^x \pmod q$, $com = g^v \pmod q$ e $resp = v - ch \cdot x \pmod q$, a prova de conhecimento é válida pois:

$$\begin{aligned} com &= g^{resp} \cdot y^{ch} \pmod q \\ g^v &= g^{v-ch \cdot x} (g^x)^{ch} \pmod q \\ g^v &= g^{v-ch \cdot x} g^{ch \cdot x} \pmod q \\ g^v &= g^v \pmod q \end{aligned}$$

Adicionalmente, é computacionalmente inviável para \mathcal{V} descobrir o valor secreto x em função dele ser oculto pela amostra aleatória v que é apenas conhecida por \mathcal{P} . □

2.2.3.1 Prova de conhecimento zero não interativa

No entanto, nem todos os casos de uso permitem a interação entre \mathcal{P} e \mathcal{V} . Consequentemente, há interesse em converter provas de conhecimento zero em provas de conhecimento zero não interativas. Esta conversão é normalmente obtida através da transformação proposta por Fiat e Shamir (1986), que exige que o protocolo seja de *public-coin*. A técnica substitui as rodadas interativas por chamadas para um oráculo randômico, na prática, uma função de resumo criptográfico é utilizada.

Considerando uma função pública de resumo criptográfico f , uma maneira simplificada de expressar o resultado da transformação de Fiat e Shamir (1986) do Exemplo 2.2.1 é:

1. \mathcal{P} amostra aleatoriamente $v \xleftarrow{\$} \mathbb{Z}_q^*$ e computa $com = g^v \pmod q$.
2. \mathcal{P} executa $ch = f(g, y, com)$.
3. \mathcal{P} computa $resp = v - ch \cdot x \pmod q$.

4. A prova de conhecimento resultante é $(com, resp)$.
5. Um \mathcal{V} qualquer a partir da prova consegue recalculuar ch e verificar a asserção $com \stackrel{?}{=} g^{resp} \cdot y^{ch} \pmod q$.

2.2.4 Esquema de comprometimento

Esquemas de comprometimento são primitivas criptográficas que permitem uma parte se comprometer com um valor e mantê-lo oculto. Goldreich (2006) compara o esquema com um envelope selado. Ao colocar uma nota nesse envelope, uma parte compromete-se com o conteúdo da nota, mas ainda o mantém em segredo.

Definição 2.2.11 (Esquema de Comprometimento). Um esquema de comprometimento é um protocolo de 2 fases que envolve uma parte chamada de *remetente*, que pode comprometer-se com um valor e um *receptor*. As fases do protocolo são:

1. **Fase de Confirmação:** o remetente é vinculado a um valor único (o comprometimento), com a fase não produzindo nenhuma informação deste valor para o receptor;
2. **Fase de Revelação:** a valor é revelado pelo remetente e o receptor verifica sua autenticidade.

Tipicamente, para o valor ser revelado durante a segunda fase é necessária uma chave chamada de abertura (*opening*), que é enviada pelo remetente durante a primeira ou a segunda fase do protocolo. O primeiro caso, pode permitir a realização da segunda fase de maneira *offline*. Já o segundo caso, é interessante quando o remetente não confia completamente no receptor (HALEVI; MICALI, 1996) que, por meio da abertura, pode em alguns casos obter o valor da mensagem antes da fase de revelação. Adicionalmente, para os dois casos, normalmente são implementadas provas de conhecimento zero para impor um comportamento honesto das partes envolvidas.

Duas principais propriedades devem ser garantidas em um esquema de comprometimento, 1) *hiding*, para que no fim da primeira fase, nenhuma informação do valor do remetente possivelmente seja vazada; e 2) *binding*, de forma que o exista apenas um único valor que o receptor na segunda fase aceite como legítimo.

2.3 CRIPTOGRAFIA PÓS-QUÂNTICA

Esquemas criptográficos atuais são fortemente dependentes de problemas denominados como clássicos. Dos algoritmos anteriormente citados, RSA baseia-se no problema de fatoração de números inteiros e ElGamal no de logaritmos discretos. Ambos tem sua segurança comprometida dado um computador quântico suficientemente poderoso aplicando o algoritmo de Shor (SHOR, 1999).

Dessa forma, criptografia pós-quântica nasceu como uma vertente da segurança digital motivada pela busca de esquemas criptográficos baseados em problemas que não são vulneráveis tanto a ataques quânticos, quanto ataques tradicionais. Essa preocupação não é recente, com estudos referentes ao impacto da computação quântica para criptografia tradicional podendo ser traçados para antes do século XXI. Notavelmente, o algoritmo de Shor (SHOR, 1999), apresentado em 1999, foi um dos grandes marcos para o surgimento dessa área.

Em 2016, o NIST, publicou um relatório (CHEN et al., 2016) sobre o atual estado da arte da computação quântica, focando em seu impacto para criptografia e a necessidade de avançar o desenvolvimento de esquemas resistentes a algoritmos quânticos. Assim, iniciaram-se esforços de pesquisa e padronização destes esquemas considerando ameaças futuras. Evidenciando o risco apresentado pela computação quântica e legitimando a importância do desenvolvimento algoritmos baseados em premissas diferentes das clássicas a Tabela 1 apresenta o impacto de um computador quântico nos principais algoritmos criptográficos da atualidade.

Tabela 1 – Impacto de computadores quânticos nos algoritmos criptográficos atuais

Algoritmo	Tipo	Impacto de um computador quântico de larga escala
AES	Chave Simétrica	Maiores tamanhos de chave necessários
SHA-2, SHA-3	Resumo Criptográfico	Maiores tamanhos de saída necessários
RSA	Chave Pública	Não mais seguro
ECDSA, ECDH (Criptografia de Curva Elíptica)	Chave Pública	Não mais seguro
DSA (Criptografia de Corpo Finito)	Chave Pública	Não mais seguro

Fonte: adaptado de (CHEN et al., 2016)

Considerando criptografia simétrica, atualmente, a principal ameaça apresentada é o algoritmo de Grover (GROVER, 1996) que é capaz de reduzir o problema de busca utilizado para adivinhação da chave de $O(N)$ para $O(\sqrt{N})$, sendo N o tamanho do espaço de chaves. Tendo como exemplo algoritmo o AES (DWORKIN et al., 2001), temos chaves de 128, 198 e 256 bits de comprimento, que tornam o tamanho do espaço de chaves 2^{128} , 2^{198} e 2^{256} respectivamente. Assim, levando em conta esta redução de complexidade, é suficiente para manter a segurança apenas o aumento do tamanho da chave¹. Para funções de resumo criptográfico, um caso semelhante se aplica e, para certos problemas, o uso de computação quântica provou-se inferior a ataques clássicos (BERNSTEIN, 2009).

Todavia, a criptografia assimétrica clássica, como anteriormente mencionado, está sujeita ao algoritmo de Shor. Considerando o problema de fatoração de inteiros como exemplo,

¹ Note que se duplicarmos o tamanho da chave, o tamanho do problema aumenta para o seu quadrado

um adversário quântico pode resolvê-lo em $O((\log N)^2 \log \log N)$ empregando o algoritmo de multiplicação de Harvey e Hoeven (2021). Assim, iniciaram-se os esforços de padronização de esquemas baseadas em premissas resistentes a algoritmos quânticos.

Em julho de 2022, NIST (NIST, 2022) divulgou os algoritmos selecionados para padronização entre todas as submissões do processo de seleção. Entre os esquemas selecionados, identifica-se grande presença de algoritmos baseados em problemas computacionais fundamentados em reticulados (BOS et al., 2018; DUCAS et al., 2018; FOUQUE et al., 2018). Adicionalmente, essas estruturas também possibilitam o desenvolvimento de primitivas criptográficas com propriedades homomórficas e esquemas de comprometimento que serão importantes para este trabalho. Assim, eles serão aprofundados na seção subsequente.

2.3.1 Reticulados

Esta seção busca expor os principais conceitos referentes a reticulados, sendo majoritariamente fundamentada pela revisão literária de Peikert et al. (2016) e o trabalho de Nguyen e Stern (2001). De forma básica, podemos definir um reticulado como um arranjo de pontos em um espaço de n dimensões sendo possível formar vetores entre eles. Há um grande interesse em problemas referentes a distância entre essas coordenadas, especialmente entre elas e a origem.

Definição 2.3.1. (Reticulado) Um reticulado \mathcal{L} de n dimensões pode ser definido como:

- Um subconjunto de \mathbb{R}^n .
- Um subgrupo aditivo: $0 \in \mathcal{L}$, e $-x, x + y \in \mathcal{L}$ para cada $x, y \in \mathcal{L}$.
- Discreto: cada $x \in \mathcal{L}$ tem uma vizinhança em \mathbb{R}^n em que x é o único ponto no reticulado.

Qualquer reticulado não-trivial é infinito, mas ele pode ser finitamente gerado com uma base. Adicionalmente, para esta seção definiremos $\lambda_i(\mathcal{L})$ como o menor vetor v de forma que \mathcal{L} tem i -ésimos vetores linearmente independentes de norma euclidiana de no máximo v .

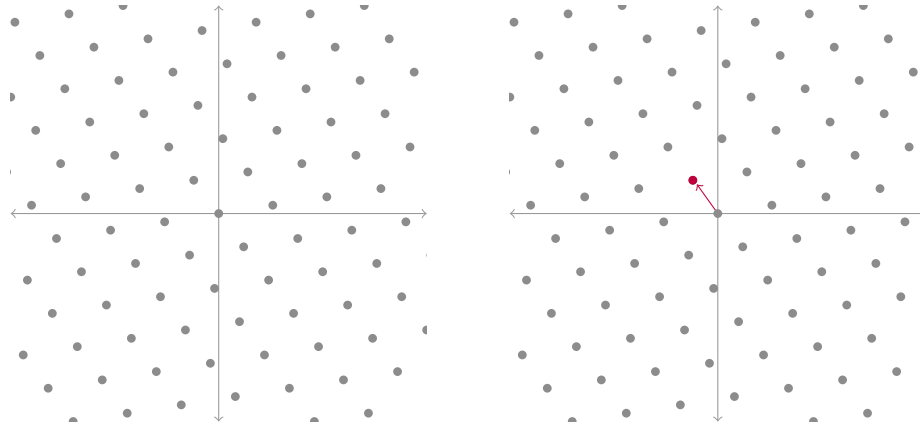
Definição 2.3.2. (Base de um reticulado)

Uma base \mathbf{B} é composta de vetores linearmente independentes b_1, b_2, \dots, b_k em um espaço euclidiano de dimensão k , que gera um reticulado finito \mathcal{L} por meio da combinação linear dos elementos de \mathbf{B} com coeficientes inteiros.

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^k = \left\{ \sum_{i=1}^k z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\}$$

O inteiro k é conhecido como o *rank* da base e quando $n = k$, o reticulado é classificado como *full-rank*.

Figura 2 – SVP para um reticulado de duas dimensões



Fonte: (JEAN, 2016)

2.3.1.1 Problemas Computacionais

Com a contextualização básica de um reticulado exposta, é possível definir os principais problemas computacionais utilizados na criptografia. É importantíssimo destacar o *Shortest Vector Problem* (SVP) (Definição 2.3.3) e o *Closest Vector Problem* (CVP) (Definição 2.3.6), que são a principal fundação para as conjecturas modernas da criptografia baseada em reticulados.

Definição 2.3.3. (SVP) Dado uma base \mathbf{B} arbitrária e um reticulado $\mathcal{L} = \mathcal{L}(\mathbf{B})$ de n dimensões, encontre o menor vetor não-zero, ou seja, um $v \in \mathcal{L}$ para qual $\|v\| = \lambda_1(\mathcal{L})$, também podendo ser representado como:

$$\lambda_1(\mathcal{L}) := \min_{v \in \mathcal{L} \setminus \{0\}} \|v\|$$

Definição 2.3.4. (*Approximate Shortest Vector Problem* (SVP $_\gamma$)) Dado uma base \mathbf{B} arbitrária e um reticulado $\mathcal{L} = \mathcal{L}(\mathbf{B})$ de n dimensões, encontre um vetor não-zero $v \in \mathcal{L}$ para qual $\|v\| \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$;

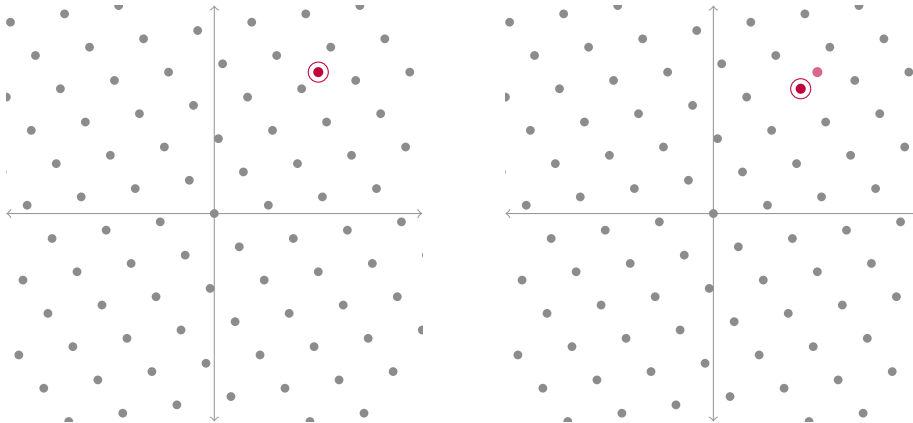
Definição 2.3.5. (*Decisional SVP* $_\gamma$) Dado uma base \mathbf{B} arbitrária e um reticulado $\mathcal{L} = \mathcal{L}(\mathbf{B})$ de n dimensões determine qual das seguintes comparações é verdade $\lambda_1(\mathcal{L}) \leq 1$ ou $\lambda_1(\mathcal{L}) > \gamma(n)$;

Definição 2.3.6. (CVP) Dado uma base \mathbf{B} arbitrária de um reticulado $\mathcal{L} = \mathcal{L}(\mathbf{B})$ de n dimensões e um vetor qualquer $v \notin \mathcal{L}$ presente em \mathbb{Z}^n , encontre o vetor u de forma que a norma euclidiana $\|u - v\|$ seja a menor possível, ou seja:

$$u := \min_{w \in \mathcal{L}} \|w - v\|$$

As figuras 2 e 3 apresentam graficamente o SVP e CVP de forma simplificada, nelas é possível observar os vetores desejáveis para cada problema em um plano bidimensional. A seguir, serão abordados os dois problemas (*Short Integer Solution* (SIS) e *Learning With Errors* (LWE)) fundamentais para formação de primitivas criptográficas resistentes tanto contra computadores quânticos quanto tradicionais.

Figura 3 – CVP para um reticulado de duas dimensões



Fonte: (JEAN, 2016)

2.3.1.2 Short Integer Solution (SIS)

O SIS é o núcleo de diversas primitivas criptográficas baseadas em reticulados, notavelmente, funções de *hash* e esquemas de assinatura digital. O problema foi introduzido por Ajtai (1996) como o desafio de encontrar uma combinação linear não-trivial suficientemente pequena e de soma 0 dado muitos elementos uniformemente aleatórios de um grande grupo finito aditivo.

Definição 2.3.7 ($SIS_{n,q,\beta,m}$). SIS pode ser parametrizado por n e q que definem o grupo \mathbb{Z}_q^n , um real positivo β e um número m de elementos do grupo. Dessa forma, dado m vetores uniformemente aleatórios $\mathbf{a}_i \in \mathbb{Z}_q^n$, formando as colunas da matriz $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, SIS impõe o desafio de encontrar um vetor de inteiros não-zero $\mathbf{z} \in \mathbb{Z}^m$ de norma euclidiana $\|\mathbf{z}\| \leq \beta$ de tal modo que:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{Az} = \sum_i \mathbf{a}_i \cdot z_i = \mathbf{0} \in \mathbb{Z}_q^n.$$

De forma que, as restrições impostas de $\|\mathbf{z}\| \leq \beta$ e $q > \beta$ garantem a não trivialidade do vetor, já que a ausência delas torna $\mathbf{z} = (q, 0, \dots, 0) \in \mathbb{Z}^m$ uma solução possível.

2.3.1.3 Learning With Errors (LWE)

Introduzido por Regev (2009), LWE é a peça fundamental para construção de primitivas criptográficas, em especial para esquemas de encriptação por chave pública, sendo possível instanciar algoritmos homomórficos com base na dificuldade de resolver esse problema. De forma geral, LWE é uma maneira de esconder um segredo a partir da adição de erros (*noise*) nele. Duas vertentes são definidas a partir do problema, busca e decisão.

Definição 2.3.8 (Distribuição LWE). LWE pode ser parametrizado por n e q que definem o grupo \mathbb{Z}_q^n e uma distribuição de erros \mathcal{X} em \mathbb{Z} que usualmente é uma gaussiana discreta de comprimento $\alpha \cdot q$ para alguns $\alpha < 1$, sendo que α pode ser chamado de taxa de erro relativa.

Assim, para um vetor $s \in \mathbb{Z}_q^n$ chamado de segredo, a distribuição $\text{LWE}_{A_{s,\mathcal{X}}}$ em $\mathbb{Z}_q^n \times \mathbb{Z}_q$ é amostrada escolhendo $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$, selecionando $e \xleftarrow{\$} \mathcal{X}$ e retornando a amostra $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e \pmod{q})$.

Definição 2.3.9 (Busca LWE). Dadas m amostras independentes $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ retiradas da distribuição $\text{LWE}_{A_{s,\mathcal{X}}}$ para um $s \xleftarrow{\$} \mathbb{Z}_q^n$ uniformemente aleatório, encontre s .

Definição 2.3.10 (Decisão LWE). Dado m amostras independentes $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ descubra para cada amostra se ela é distribuída de acordo com $A_{s,\mathcal{X}}$ para um $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ ou de acordo com uma distribuição uniforme.

2.3.1.4 Ring-SIS

Análogo ao problema de SIS, Micciancio (2007) apresentou a variação *Ring-SIS* caracterizada como um problema computacional que é capaz de instanciar algoritmos mais compactos e eficientes por trocar a estrutura algébrica central de um grupo para um anel.

Definição 2.3.11 ($\text{R-SIS}_{q,\beta,m}$). *Ring-SIS* é parametrizado por um anel R , normalmente definido como um anel polinomial de grau n de forma que $R = \mathbb{Z}[X]/(f(X))$, um inteiro positivo q que é o módulo do anel $R_q = R/q$, um limite de norma real $\beta > 0$ e um número m de amostras. Dessa forma, dadas m amostras uniformes de R_q que definem o vetor $\vec{a} \in R_q^m$, o desafio de R-SIS é encontrar um vetor não-zero $\vec{z} \in R^m$ de norma $\|\vec{z}\| \geq \beta$ de forma que:

$$f_{\vec{a}}(\vec{z}) := \langle \vec{a}, \vec{z} \rangle = \vec{a} \cdot \vec{z} = \sum_i a_i \cdot z_i = 0 \in R_q.$$

2.3.1.5 Ring-LWE

Lyubashevsky et al. (2013) apresentaram o problema baseado em anéis análogo ao LWE . Semelhante a R-SIS , este problema é a base para esquemas superiores nos quesitos de compactidade e eficiência em relação a sua contrapartida.

Definição 2.3.12 (Distribuição *Ring-LWE*). *Ring-LWE* pode ser parametrizado por um anel R de grau n em \mathbb{Z} , um inteiro positivo q que é o módulo do anel quociente $R_q = R/q$ e uma distribuição de erro \mathcal{X} em R , normalmente definida como uma Gaussiana discreta. Assim, para um segredo aleatório $s \xleftarrow{\$} R_q$, a distribuição $\text{R-LWE}_{A_{s,\mathcal{X}}}$ sobre $R_q \times R_q$ é amostrada escolhendo um $a \xleftarrow{\$} R_q$, selecionando um $e \xleftarrow{\$} \mathcal{X}$ e retornando $(a, b = s \cdot a + e \pmod{q})$.

Definição 2.3.13 (Decisão $\text{R-LWE}_{q,\mathcal{X},m}$). Dado m amostras independentes $(\mathbf{a}_i, b_i) \in R_q \times R_q$ descubra para cada amostra se ela é distribuída de acordo com $A_{s,\mathcal{X}}$ para um $s \xleftarrow{\$} R_q$ ou de acordo com uma distribuição uniforme.

2.4 CRIPTOGRAFIA HOMOMÓRFICA

Na criptografia, homomorfismo é uma propriedade que possibilita a realização de operações em textos cifrados sem decifração preliminar. Tradicionalmente, o processo de encriptação garante confidencialidade a informações por meio da geração de textos cifrados alheios ao conteúdo original, impossibilitando operações sobre os dados cifrados. Dessa forma, aplicações devem decifrar a informação antes de operar sobre ela. Isso, evidentemente, introduz problemas de privacidade, especialmente quando dada informação deve transitar em canais inseguros ou ser acessada por terceiros.

Assim, a motivação de encriptação homomórfica nasceu como solução para aplicações que demandam tanto a privacidade quanto a necessidade de operar facilmente sobre os dados. Podemos ver essa propriedade sendo aproveitada em aplicações de votação eletrônica (ADIDA et al., 2023); recuperação de informações privadas (KUSHILEVITZ; OSTROVSKY, 1997); *machine learning* (SUN et al., 2020); e dados de saúde (DOWLIN et al., 2017).

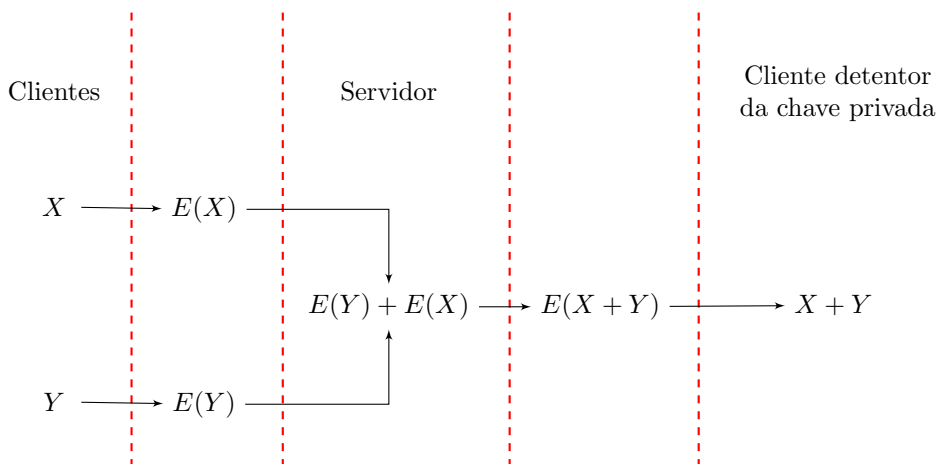
Definição 2.4.1. Um esquema criptográfico é considerado homomórfico sobre uma operação $*$ se ele suporta determinada equação:

$$E(m_1) * E(m_2) = E(m_1 * m_2), \forall m_1, m_2 \in M$$

onde E é a operação de encriptação e M o conjunto de todas as mensagens possíveis (ACAR et al., 2018).

É possível ilustrar estas propriedades por meio do fluxo apresentado na Figura 4. Tendo como exemplo a operação de adição, o servidor em nenhum momento tem noção do valores de X ou Y , mas ainda é capaz calcular a soma destes números por meio de seus textos cifrados. Assim, apenas o cliente detentor da chave privada saberá o resultado final.

Figura 4 – Exemplo de aplicação para *Homomorphic Encryption* (HE)



Fonte: autoria própria.

Rivest et al. (1978a) introduziram *Homomorphic Encryption* (HE) como um problema em aberto. A partir deste ano e nas décadas subsequentes, diversos esquemas criptográficos de-

envolvidos apresentavam propriedades homomórficas sobre algumas operações, sendo agrupados como *Partially Homomorphic Encryption* (PHE). Alguns exemplos notáveis são RSA (RIVEST et al., 1978b) e ElGamal (ELGAMAL, 1985) para a operação de multiplicação e Benaloh (BENALOH, 1994) para adição.

Estes esquemas já são capazes de atender as necessidades de algumas aplicações (*e.g.* votação eletrônica), mas equações de maior complexidade demandam uma maior variedade de operações possíveis. *Somewhat Homomorphic Encryption* (SWHE) atende essas necessidades de forma restrita, suportando tanto adição e multiplicação mas em quantidade limitada. Essa restrição deve-se ao crescimento de um parâmetro de ruído que deve ser adicionado a medida que as operações são realizadas para manter a encriptação única, preservando a segurança do esquema.

Mais de três décadas após a introdução do problema, Gentry (2009) propôs tanto o primeiro esquema de *Fully Homomorphic Encryption* (FHE) quanto uma estrutura para o desenvolvimento de um esquema de FHE. Este, baseado em reticulados ideais, provê tanto adição quanto multiplicação sem restrições. Para obter essa propriedade, Gentry apresenta a conversão de um sistema SWHE para um FHE, diminuindo o ruído por meio da operação de *bootstrapping* quando ele torna-se grande demais.

2.4.1 Homomorfismo de ElGamal

Como anteriormente comentado na Subseção 2.2.2, o esquema ElGamal, entre os algoritmos pré-quânticos é de maior importância para o trabalho, decorrente do aproveitamento de suas propriedades homomórficas no protocolo de votação eletrônica de Cramer et al. (1997). Dessa forma, baseando-se no Protocolo 1, será demonstrado o homomorfismo do esquema.

Proposição 1. O esquema ElGamal é homomórfico para operação de multiplicação, de forma que para duas mensagens M_1 e M_2 , $E(M_1) \cdot E(M_2) = E(M_1 \cdot M_2)$.

Demonstração. Dado duas mensagens (M_1, M_2) que são encriptadas respectivamente com dois números selecionados aleatoriamente k_1 e k_2 e com a mesma chave pública y temos:

$$\begin{aligned} a_1 &= g^{k_1} \pmod p & b_1 &= y^{k_1} M_1 \pmod p \\ a_2 &= g^{k_2} \pmod p & b_2 &= y^{k_2} M_2 \pmod p. \end{aligned}$$

Que correspondem a duas cifras $E(M_1) = (a_1, b_1)$ e $E(M_2) = (a_2, b_2)$. O produto entre elas é representado por $E(M_1) \cdot E(M_2) = (a_1 a_2, b_1 b_2) = (g^{k_1+k_2}, y^{k_1+k_2} M_1 M_2)$. Assim, é possível decriptar $M_1 \cdot M_2$ com a cifra $(a_1 a_2, b_1 b_2)$.

$$\begin{aligned}
D(E(M_1) \cdot E(M_2)) &= \frac{b_1}{a_1^x} \cdot \frac{b_2}{a_2^x} \pmod{p} \\
&= \frac{y^{k_1} M_1 y^{k_2} M_2}{(g^{k_1})^x (g^{k_2})^x} \pmod{p} \\
&= \frac{(g^x)^{k_1} M_1 (g^x)^{k_2} M_2}{g^{k_1 x} g^{k_2 x}} \pmod{p} \\
&= \frac{g^{k_1 x} M_1 g^{k_2 x} M_2}{g^{k_1 x} g^{k_2 x}} \pmod{p} \\
&= M_1 M_2 \pmod{p}.
\end{aligned}$$

□

Proposição 2. Sejam M_1 e M_2 duas mensagens. É possível executar a adição homomórfica encriptando g^{M_1} e g^{M_2} ao invés de M_1 e M_2 e realizando a multiplicação proposta na Proposição 1.

Demonstração.

$$\begin{aligned}
D(E(g^{M_1}) \cdot E(g^{M_2})) &= \frac{b_1}{a_1^x} \cdot \frac{b_2}{a_2^x} \\
&= \frac{g^{k_1 x} g^{M_1} g^{k_2 x} g^{M_2}}{g^{k_1 x} g^{k_2 x}} \\
&= g^{M_1 + M_2} \\
M_1 + M_2 &= \log_g g^{M_1 + M_2}.
\end{aligned}$$

□

2.5 VOTAÇÃO ELETRÔNICA

Historicamente, as eleições se traduzem como um mecanismo que avalia o poder de escolha de um determinado grupo e, com a ininterrupta digitalização do mundo, é lógico modernizá-las para melhor adequá-las as comodidades oferecidas pela tecnologia da mesma forma que diversos outros processos foram avançados, por exemplo, *e-banking* e *e-commerce*. No entanto, votação é uma tarefa distinta, de forma que a disponibilidade de dados é um aspecto severamente limitante em função de seus altíssimos requisitos de privacidade.

Ainda considerando os exemplos de *e-banking* e *e-commerce*, esses sistemas esperam a ocorrência de fraude. Não é incomum casos de cartões de créditos clonados ou transações fraudulentas, assim, esses custos são absorvidos pelas organizações causando bilhões em prejuízo anualmente, no entanto, essas perdas não se comparam com os danos causados por fraudes em eleições.

Diversos fatores, especialmente humanos, devem ser considerados para votações, venda de votos e coação são alguns dos aspectos que tornam o processo tão delicado e dependente da confidencialidade. Assim, é imposto um desafio de balancear o sigilo de um voto, com garantias

da honestidade da eleição. Entretanto, mesmo em condições sublimes, não é possível prover a um votante a quantidade ideal de confiança que garanta a integridade de seu voto em processos tradicionais.

No prefácio de (HAO; RYAN, 2016), apresentado por Benaloh, é comentado que qualquer participante de uma eleição é potencialmente malicioso. Considerando essas palavras, é compreensível o interesse por acessar ou adulterar o resultado dos votos, assim, a desonestidade e antagonismo dos votantes, oficiais, candidatos, auditores e equipamentos são fatores que devem ser esperados e mitigados em um protocolo de votação.

No âmbito digital, Chaum (1981) introduziu o uso de criptografia para eleições, apresentando primitivas criptográficas para segurança durante comunicações anônimas. Em geral, esquemas de votação eletrônica cifram cédulas para ocultar o valor dos votos. Notavelmente, o desafio fundamental desses protocolos é principalmente o cálculo da contagem sem quebrar a privacidade dos eleitores e, ao mesmo tempo, conseguir provar que todos os votos foram contados corretamente. Para resolver esse problema, existem três estruturas principais:

1. *contagem homomórfica*: todas as cédulas cifradas são somadas e depois decifradas, então, nenhum voto singular é revelado;
2. *assinatura cega*: as cédulas são anonimizadas antes de serem enviadas;
3. *mix-nets*: as cifras são embaralhadas de uma maneira que não existe nenhuma correlação entre uma cédula cifrada e o voto que ela contém.

Desde a contribuição acadêmica de Chaum (1981), o tema de votação eletrônica foi sujeito a avanços até atingir maior maturidade, especialmente na definição dos fundamentos essenciais que constituem um esquema padronizado e seguro. Assim, definiu-se propriedades necessárias ou desejáveis para protocolos de votação eletrônica que devem ser seguidos para o uso em aplicações fora do âmbito acadêmico.

2.5.1 Propriedades fundamentais

Mesmo não inteiramente padronizado, é possível abstrair a maioria dos sistemas eleitorais em quatro principais tarefas (CRANOR; CYTRON, 1996), que devem ser propriamente executadas para garantia da confiabilidade do sistema, elas são:

1. **Cadastro**: envolve a coleta de uma lista de entidades elegíveis para votar.
2. **Validação**: verificação das credenciais daqueles que tentam votar, permitindo o voto apenas daqueles que são elegíveis e ainda não votaram.
3. **Coleta**: envolve a coleta das cédulas de votação.
4. **Contagem**: envolve a contagem de votos e revelação do resultado.

Como anteriormente citado, corrupção é a maior preocupação na construção de um sistema de votação, cada tarefa anteriormente apresentada dispõe de oportunidades de comportamento adversário pelas diversas entidades presentes. Assim, medidas devem ser tomadas para inibir adulterações ou outros comportamentos desonestos, especialmente quando essas tarefas devem ser executadas de forma eletrônica que carecem de algumas medidas físicas presentes em eleições tradicionais.

Cranor e Cytron (1996) uniram diversas abordagens de sistemas eletrônicos de votação apresentados na literatura e identificou quatro propriedades centrais que são essenciais em quase todos os sistemas eleitorais:

- **Precisão:** o sistema é preciso se um voto válido é inalterável e não pode ser eliminado da contagem, enquanto um voto inválido não pode de forma alguma ser considerado para o resultado final.
- **Democracia:** o sistema é democrático se permite apenas votos de entidades elegíveis e garante que uma entidade apenas vote uma vez.
- **Privacidade:** o sistema tem privacidade se não é possível para qualquer entidade ligar um votante a sua cédula de votação e nenhum votante é capaz de provar que votou de forma particular, ou seja, em casos de venda de votos, por exemplo, não é possível provar que o voto realmente foi destinado ao candidato desejado pelo comprador.
- **Verificabilidade:** um sistema é verificável se um votante pode independentemente verificar se seu voto foi corretamente contado. Normalmente é desejável verificabilidade individual, que permite que um votante verifique que seu voto foi computado, e verificabilidade universal, que garante a legitimidade da contagem final de votos.

Definição 2.5.1. (End-to-End Verifiable (E2E-V)) Um sistema de votação eletrônica é considerado E2E-V se ele provê técnicas para votantes individualmente verificarem aspectos cruciais do resultado da eleição, sem a necessidade dos votantes confiarem no *software*, *hardware*, oficiais da eleição, procedimento ou observadores (BENALOH et al., 2015).

Outros trabalhos também citam robustez, auditabilidade, certificabilidade, responsabilidade e divulgabilidade de sistema, entre outros (JR, 2000; NEUMANN, 1993; SHAMOS, 1993). Em geral, muitos critérios também são dependentes da instituição organizadora da eleição, por exemplo, a *International Association for Cryptologic Research (IACR)* adicionalmente demanda que o software seja de código aberto (IACR, 2008).

No entanto, uma propriedade que votação eletrônica é incapaz de atender é a transparência (HAO; RYAN, 2016). Enquanto é simples o entendimento e monitoramento por um votante do processo eleitoral por papel, a digitalização do protocolo, envolvendo a inserção de garantias criptográficas e a contagem por computadores que caracterizam um processo que só pode ser compreendido por técnicos. Dessa forma, isso constitui uma carência fundamental na votação eletrônica que pode gerar demais problemas que devem ser tratados.

2.5.2 Esquema multi-autoridade de Cramer et al. (1997)

Cramer et al. (1997) propõem um esquema de votação secreta multi-autoridade para eleições que busca atender os requisitos de privacidade, verificabilidade universal e robustez previamente estabelecidos na literatura em combinação com a preocupação em alcançar um desempenho ideal para eleições de larga escala, com a principal consideração sendo o nível de esforço por parte do votante para realizar o protocolo.

O esquema baseia-se em trabalhos prévios de Benaloh et al. (1986, 1987, 1994) com as partes ativas sendo l votantes $V_1 \dots V_l$ e n autoridades de contagem $A_1 \dots A_n$. Adicionalmente, é empregado um *bulleting board* BB que é o modelo de comunicação do protocolo, toda informação que passa por ele é pública, inalterável e irremovível, com qualquer parte, incluindo um observador passivo, podendo ler o conteúdo.

Em especial, o *bulletin board* tem a função de garantir a verificabilidade universal. Nele, os votantes postam suas cédulas e, decorrente das propriedades homomórficas do esquema criptográfico utilizado, a contagem final pode ser obtida e verificada em relação ao produto de todos os votos submetidos.

Para minimizar a tarefa do votante, o processo limita-se a um envio de uma única mensagem encriptada (cédula) acompanhada de uma prova de validade que indica se dada mensagem contém um voto legítimo. Adicionalmente o esquema conta com uma prova de conhecimento adicional para cada autoridade durante a contagem de votos. Ambas as provas serão abordadas em maior detalhe na Subsubseção 2.5.2.1.

Como anteriormente comentado, o esquema empregado é o ElGamal. Cramer et al. (1997) baseiam-se no criptossistema de Pedersen (1991) para gerar o protocolo considerando a ausência de uma parte confiável, sendo necessário um esforço coletivo de um subconjunto das partes envolvidas para a execução das tarefas. Em seguida, será descrito esse criptossistema para melhor detalhar o funcionamento do protocolo de votação eletrônica.

Protocolo 2. Criptossistema ElGamal para multi-autoridades de Pedersen (1991)

Os passos principais do protocolo são *geração de chaves*, *cifragem* e *decifragem*, sendo construídos em cima do esquema (t, n) -*threshold* de Shamir para *secret sharing* (SHAMIR, 1979), de forma que um conjunto de n autoridades necessite de apenas t entidades ($1 \leq t \leq n$) para reconstituir o segredo e decifrar um texto cifrado. O seguinte processo tem a principal função de garantir a robustez, de forma que nenhum participante do protocolo tenha conhecimento da chave secreta.

1. Geração de chaves $K(\lambda, A)$.

Entrada: λ , conjunto de autoridades A de tamanho n .

Saída: par de chaves (sk, pk) .

- 1) cada autoridade A_j executa a geração de chaves proposta no Protocolo 1 sobre o mesmo grupo $G = \langle g \rangle$ de ordem p , resultando em um par de chaves (x_j, y_j) .

- 2) cada autoridade A_j compromete-se com sua parte x_j publicando sua chave pública $y_j = g^{x_j}$.
- 3) as partes x_j comportam-se de forma que é possível reconstituir $sk = x$ com qualquer conjunto Γ de t partes utilizando coeficientes de Lagrange apropriados:

$$x = \sum_{j \in \Gamma} x_j \gamma_{j,\Gamma}, \quad \gamma_{j,\Gamma} = \prod_{l \in \Gamma \setminus \{j\}} \frac{l}{l-j}. \quad (2.1)$$

- 4) a chave pública $pk = y = g^x$ é publicada para todos os participantes do sistema.

2. Cifragem $E(pk, M)$.

Entrada: chave pública $pk = y$, mensagem $M \in G$.

Saída: texto cifrado C .

- 1) com a chave pública pk é encriptado M de acordo com Protocolo 1, obtendo sua cifra $C = (a, b) = (g^k, y^k M)$.

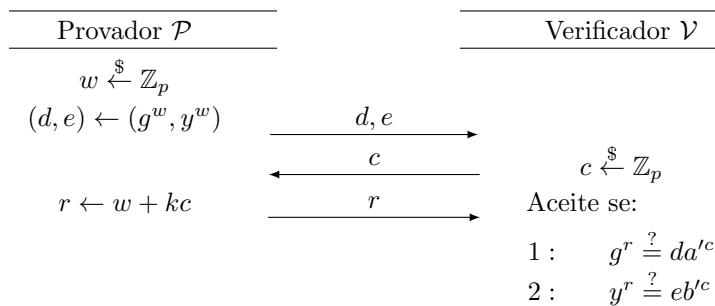
3. Decifragem $D(C, A)$.

Entrada: texto cifrado $C = (a, b) = (g^k, y^k M)$, conjunto de autoridades A de tamanho n .

Saída: mensagem $M \in G$.

- 1) Cada autoridade A_j transmite $w_j = a^{x_j}$ e prova por conhecimento zero que $\log_g y_j = \log_a w_j$ por meio do protocolo- Σ apresentado na Figura 5 considerando como informações públicas o gerador g , a chave pública y , e a tupla $(a', b') = (g^k, y^k)$. Adicionalmente, nota-se que o valor de k não pode ser revelado.

Figura 5 – Protocolo- Σ de conhecimento-zero para $\log_g y_j = \log_a w_j$



Fonte: adaptado de (CRAMER et al., 1997)

- 2) Seja Γ o subconjunto de autoridades que passaram na prova de conhecimento-zero. Na condição do tamanho de Γ ser maior ou igual a t e lembrando que no esquema tradicional do Protocolo 1 a mensagem é $M = \frac{b}{a^x}$, é possível recuperar M de maneira similar elevando a a ambos os lados da Eq. (2.1), mesmo que $n - t$ autoridades sejam maliciosas ou falhem durante o protocolo:

$$M = b / \prod_{j \in \Gamma} w_j^{\gamma_j, \Gamma}. \quad (2.2)$$

Demonstração. Elevando a a ambos os lados da Eq. (2.1) obtemos:

$$\begin{aligned} a^x &= a^{\sum_{j \in \Gamma} x_j \gamma_j, \Gamma} \\ &= \prod_{j \in \Gamma} (a^{x_j})^{\gamma_j, \Gamma} \\ &= \prod_{j \in \Gamma} w_j^{\gamma_j, \Gamma}. \end{aligned}$$

Assim, é possível obter uma representação válida de M por meio da Eq. (2.2) realizando os seguintes passos.

$$\begin{aligned} D(C) &= b / \prod_{j \in \Gamma} w_j^{\gamma_j, \Gamma} \\ &= b / a^x \\ &= M. \end{aligned}$$

□

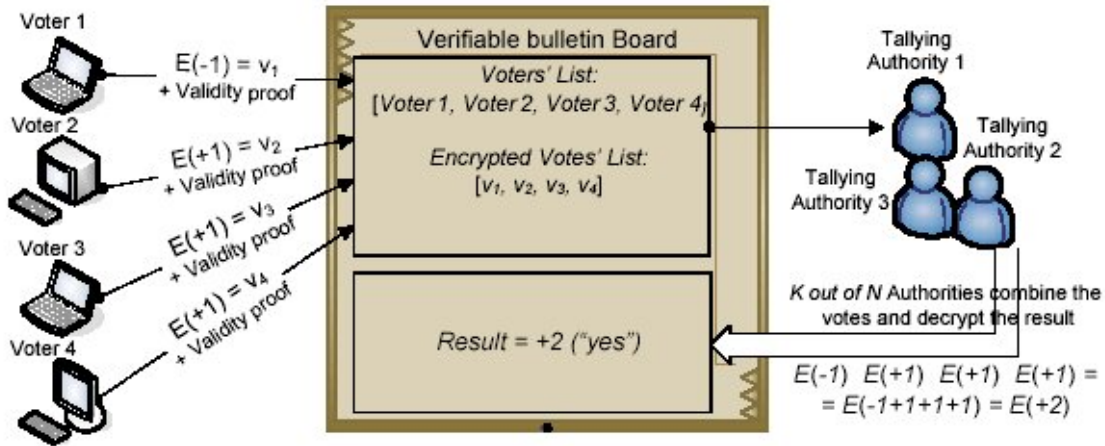
O esquema é inicialmente construído para eleições onde o votante enviará uma cifra ElGamal de uma mensagem M com apenas duas opções. Para melhor entendimento, consideraremos essas opções como uma resposta de sim e não, M_1 e M_0 respectivamente, logo $M \in \{M_0, M_1\}$. Suplementarmente, é aproveitável a propriedade homomórfica para a operação de adição na Proposição 2, assim, $M_0 = g^{-1}$ e $M_1 = g$. Essa escolha será justificada na contagem dos votos.

Com tal característica, o fluxo básico de funcionamento do protocolo, considerando as quatro principais tarefas de votação eletrônica anteriormente abordadas, é descrito como:

1. **Cadastro:** são cadastradas n autoridades e l votantes.
2. **Votação:** Votante V_i cifra seu voto, enviando a cédula encriptada (a_i, b_i) para o *bulletin board* acompanhado da prova de validade que será detalhada na Subsubseção 2.5.2.1.
3. **Coleta:** Quando a votação é encerrada, as provas de validade são verificadas pelas autoridades e o produto $(a, b) = (\prod_{i=1}^l a_i, \prod_{i=1}^l b_i)$ é formado. Por conta da Proposição 1, (a, b) é uma encriptação válida para o produto de todos os votos.
4. **Contagem:** As autoridades executam a decriptação conjunta de (a, b) , proposta no Protocolo 2, incluindo a prova de conhecimento zero, para obter o valor de $W = \frac{b}{a^x}$.

Como a mensagem M de um único voto é g ou g^{-1} e W é o produto de todos os votos, W pode ser lido como g^T , sendo T a diferença entre os votos de sim e os de não, $-l \leq T \leq l$. Consequentemente $T = \log_g W$, que é difícil computar. Todavia, como l é relativamente

Figura 6 – Protocolo multi-autoridade homomórfico de Cramer



Fonte: (MAGKOS et al., 2007)

pequeno, o valor de T pode ser descoberto com $O(l)$ multiplicações modulares, iterativamente calculando $g^{-l}, g^{-l+1}, g^{-l+2}, \dots$ até W ser encontrado.

É possível observar o fluxo apresentado acima pela Figura 6, nela, cada votante envia remotamente uma representação encriptada de 1 ou -1. O *bulletin board* armazena as listas de votantes cadastrados e votos encriptados, que, finalmente, são enviados para as autoridades. Considerando um número apropriado de autoridades que passaram na prova de conhecimento, é decriptado o produto entre os votos. Por fim, neste caso, o resultado foi +2, indicando uma diferença dois votos a favor de sim.

2.5.2.1 Provas de validade

A prova de conhecimento-zero supracitada na decriptação do Protocolo 2 é referente ao protocolo de três movimentos Chaum-Pederson (CHAUM; PEDERSEN, 1992) para igualdade entre logaritmos discretos, considerando a mesma anotação previamente apresentada e a Figura 5, um provador mostra posse de um $k \in \mathbb{Z}_p$ que satisfaça $a = g^k$ e $b = y^k$ que é uma cifra válida para uma mensagem $M \equiv 1 \pmod p$. Podemos representar essa comunicação como uma tupla (d, e, c, r) .

Originalmente, Chaum-Pederson não é de conhecimento-zero, mas Cramer garante essa propriedade sobre um verificador honesto em função de que para um desafio c e r aleatórios, temos que uma comunicação $(g^r a^{-c}, y^r b^{-c}, c, r)$ também é válida.

Demonstração. Considerando uma comunicação válida $C_1 = (d, e, c, r)$, temos as variáveis:

$$\begin{aligned} a &= g^k & b &= y^k \\ d &= g^w & e &= y^w. \end{aligned}$$

Assim, para uma nova comunicação $C_2 = (g^r a^{-c}, y^r b^{-c}, c, r)$ temos:

$$\begin{aligned} a^{-c} &= g^{-kc} \\ g^r a^{-c} &= g^{r-kc} \\ b^{-c} &= y^{-kc} \\ y^r b^{-c} &= y^{r-kc}. \end{aligned}$$

Como $r = w + kc$, podemos substituí-lo na fórmula para obter (d, e) a partir de $(g^r a^{-c}, y^r b^{-c})$.

$$\begin{aligned} g^r a^{-c} &= g^{w+kc-kc} = g^w = d \\ y^r b^{-c} &= y^{w+kc-kc} = y^w = e. \end{aligned}$$

Assim, a comunicação C_2 é igual a (d, e, c, r) , então, $C_1 = C_2$. \square

Para a prova de validade de um votante, considerando que um voto (x, y) representa a encriptação de uma mensagem $M \in \{M_0, M_1\}$. Um votante conhece o valor de M e um verificador conhece os valores possíveis para M , assim, deve ser provado que (x, y) é um texto cifrado que corresponde a um desses valores sem revelar o valor real de M que é dado por:

$$\log_g a = \log_y(b/M_0) \vee \log_g a = \log_y(b/M_1).$$

Para representar essas duas igualdades logarítmicas é empregado o protocolo de Chaum e Pedersen (1992) novamente em conjunto com técnicas apresentadas por Cramer et al. (1994) para sistemas de prova interativas. A Figura 7 apresenta esta prova, no qual temos duas possibilidades de execução para um provador \mathcal{P} , quando o voto v é 1 ou -1 . No primeiro caso, temos os valores do voto inseridos em α_1 e β_1 , sendo acompanhados de α_2 e β_2 que são indistinguíveis de um voto válido, o contrário acontece na outra situação com o voto em α_2 e β_2 . Assim, a igualdade logarítmica pode ser verificada sem o vazamento de qualquer informação que compromete a privacidade do voto.

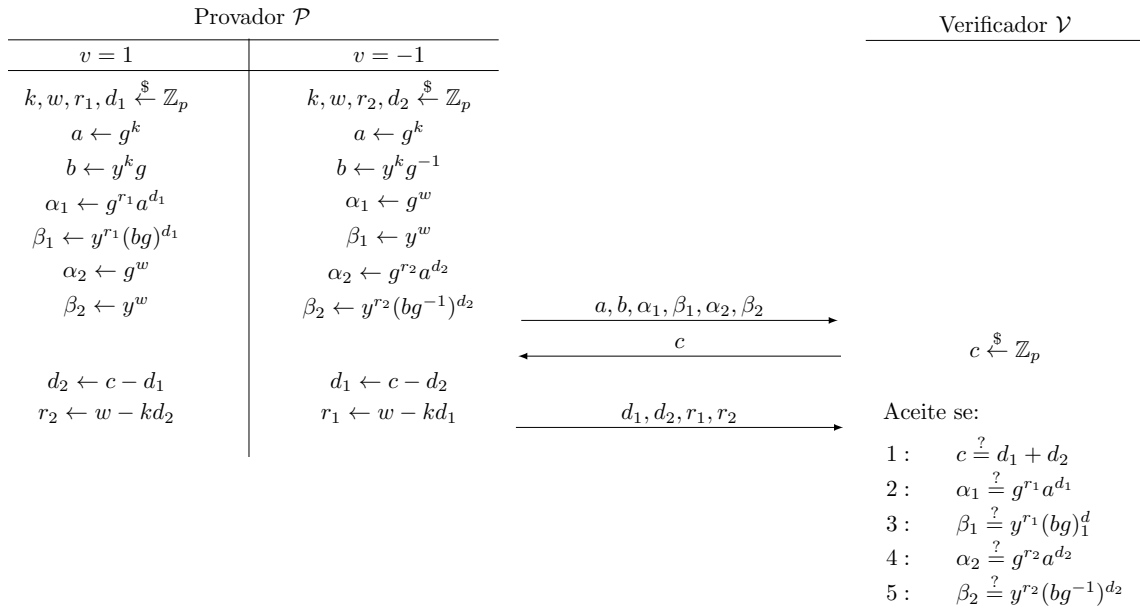
2.5.2.2 Eleições de múltipla escolha

Nem todas as eleições limitam-se a apenas 2 escolhas. O artigo propõe um método para abordar essa necessidade que não muda o tamanho das cédulas, mas aumenta o tamanho da prova. Considerando uma escolha entre K possibilidades, são pegos K geradores independentes $g_i, 1 \leq i \leq K$, e os votos são acumulados separadamente para cada opção. A prova de validade da cédula torna-se:

$$\log_g a = \log_y(b/g_1) \vee \dots \vee \log_g a = \log_y(b/g_K).$$

2.5.3 Helios

Helios (ADIDA et al., 2023) é um sistema de votação eletrônica E2E-V de código aberto. Sua primeira versão foi apresentada por Adida em 2008 (ADIDA, 2008), adotando o

Figura 7 – Protocolo- Σ de conhecimento-zero para encriptação da cédula

Fonte: adaptado de (CRAMER et al., 1997)

modelo Sako-Killian (SAKO; KILIAN, 1995) baseada em embaralhamento para computação dos votos. Nas versões subsequentes, essa implementação foi abandonada (PEREIRA, 2016) para seguir o modelo de Cramer et al. (1997) supracitado.

A aplicação ideal para Helios são cenários que priorizam integridade e privacidade mas o risco de coerção é baixo (ALI; MURRAY, 2016). Para diversos comportamentos adversários, a aplicação implementa adaptações para contorná-los, por exemplo, a técnica de *challenge-or-cast* de Benaloh (2007) para mitigar dispositivos de votação maliciosos, todavia existem pouquíssimos meios no Helios para contornar o problema coerção, especialmente quando também deseja-se obter conveniência e usabilidade para os votantes.

Um desses meios, é a capacidade de um votante de sobrescrever o voto coagido. Contudo, esta abordagem apenas transfere a responsabilidade de privacidade do envio de votos para o cadastro de votantes, não sendo considerada uma solução para o problema (ADIDA, 2008). Um esquema à prova de coerção demanda interações verdadeiramente privadas.

Contudo, essa limitação não é obstáculo para diversas aplicações. Algumas das instituições citadas por Pereira (2016) que utilizam Helios são: IACR (BENALOH et al., 2010), Universidade de Princeton (PRINCETON, 2019) e Universidade Católica da Lovaina (ADIDA et al., 2009). Nacionalmente, são vários os casos de uso que foram compilados em um excerto divulgado pela Universidade de Pelotas.

No Brasil, pode-se citar alguns exemplos de Instituições de Ensino Superior que usam o sistema Helios em eleições internas, tais como: Universidade de São Paulo (USP), Universidade de Campinas (UNICAMP), Universidade Federal de Santa Catarina (UFSC), Universidade Federal de Minas Gerais (UFMG), Universidade Federal da Grande Dourados (UFGD), Universidade

Federal de São Carlos (UFSCar). Helios é usado também por Institutos Federais, tais como: Instituto Federal do Pará (IFPA), Instituto Federal de Goiás (IFG), Instituto Federal de Rondônia (IFRO), Instituto Federal de Minas Gerais (IFMG), Instituto Federal de Santa Catarina (IFSC), Instituto Federal Fluminense (IFF), entre outros. Organizações públicas como Defensoria Pública da União e Tribunal de Justiça de Minas Gerais além de sociedades científicas como Sociedade Brasileira de Computação (SBC) e a Associação Brasileira de Métodos Computacionais em Engenharia (ABMEC) (Universidade Federal de Pelotas, 2018).

De forma complementar, é amplamente utilizada pelos exemplos acima, ao invés da versão oficial do Helios, a variação implementada pelo IFSC (CHAVES, 2023), que conta com diversas adições, incluindo autenticação via LDAP e Shibboleth, tradução de interfaces para português, ambiente para administradores e aperfeiçoamentos nas páginas web. Adicionalmente, é interessante mencionar as variações Helios Lib (ANARVOTE, 2018), que torna a aplicação independente de um sistema web e Belenios (GLONDU, 2023), construído encima do protocolo generalizado Helios-C (CORTIER et al., 2013).

2.5.4 Outros protocolos

A Noruega realizou em 2011 e 2013 eleições eletrônicas, locais e parlamentares respectivamente, nas quais os cidadãos podiam votar de casa (GJØSTEEN; LUND, 2016). A principal contribuição do sistema proposto é a implementação de códigos de retorno para votantes verificarem que suas cédulas foram computadas, trabalhando para detecção de fraudes geradas por dispositivos de votação maliciosos.

Iniciado em 2005 (MADISE; MARTENS, 2006), a Estônia atingiu grande maturidade no quesito de votação eletrônica, o protocolo baseia-se no uso do cartão de identidade nacional mandatório para todos os habitantes. Ele detém dois pares de chaves RSA e certificados ligados as chaves públicas (CLARKE; MARTENS, 2016). O primeiro par é utilizado para autenticação e o segundo para assinar digitalmente. Assim, um voto é encriptado com a chave pública da urna, mas assinado com o segundo par de chaves do cartão.

3 TRABALHOS RELACIONADOS

Votação eletrônica considerando computadores quânticos como adversários é um tema emergente, poucos trabalhos apresentam soluções para o tema que limitam-se a provas de conceito com pouca maturidade. Contextualizando brevemente o problema, atualmente, protocolos de votação eletrônica são dependentes de premissas clássicas para garantir a privacidade e verificabilidade das eleições, ou seja, a segurança dos atuais esquemas pode ser comprometida com o advento da computação quântica.

Assim, serão apresentados os principais trabalhos de votação eletrônica pós-quântica que cumprem os requisitos de serem baseados em reticulados e apresentarem um fluxo de tarefas semelhante ao apresentado por Cramer et al. (1997). Os conceitos dos artigos serão apresentados superficialmente, sendo que um deles será escolhido para detalhamento e implementação no sistema do Helios.

3.1 AN HOMOMORPHIC LWE BASED E-VOTING SCHEME

Chillotti et al. (2016) propõem o primeiro esquema de votação eletrônica pós-quântico, sendo homomórfico e baseado em LWE e SIS, com o protocolo fortemente inspirado no Helios. A principal vantagem apresentada corresponde a flexibilidade do esquema FHE que dispensa a necessidade de provas de conhecimento para a validade de um voto ou a exatidão do resultado final. No entanto, é possível inferir, mesmo na falta de uma implementação do esquema, que esse fator compromete a performance, decorrente do *bootstrapping* demandado pelo algoritmo FHE.

As partes ativas do protocolo são praticamente idênticas ao modelo de Cramer et al. (1997) com a adição de uma autoridade adicional responsável pelo registro de votantes. Adicionalmente, o artigo generaliza o esquema de forma que não é definido um criptossistema de chaves públicas específico, como o ElGamal é para o de Cramer et al. (1997). Todavia, define-se que o esquema selecionado deve ser empregado para gerar um par de chaves para cada autoridade, votante e o próprio *bulletin board*.

Um voto é representado como uma sequência encriptada de 0s e 1s que é assinada pelo votante em conjunto com as suas credenciais e enviada para o *bulletin board*, garantindo a autenticidade. Adicionalmente, o esquema permite uma maior expressividade no texto cifrado, com a encriptação baseada no *bootstrapping* de Ducas e Micciancio (2015) possibilitando transformações de textos cifrados de domínio completo para adequá-los a uma semântica específica.

No domínio da criptografia, o artigo depende especialmente do conceito de encriptação homomórfica baseada em LWE, proposta na tese de Gentry (2009), sendo apresentada uma variação do problema baseada em anéis. Adicionalmente, são aplicados conceitos de *trapdoor lattice*, que dispensam a última prova de conhecimento na verificação da contagem de votos, encriptação não-maleável e assinaturas existencialmente infalsificáveis.

Contrário ao modelo de Cramer et al. (1997), são necessárias todas as autoridades para

reconstrução da chave secreta em razão do esquema realizar uma concatenação dos sistemas LWE no lugar de um método de *secret sharing* como o de Shamir (1979). Isto torna o sistema seguro desde que pelo menos uma autoridade seja honesta, em contrapartida, o esquema torna-se limitado caso alguma autoridade recuse-se a executar a contagem.

3.2 PRACTICAL QUANTUM-SAFE VOTING FROM LATTICES

Pino et al. (2017) propõem o esquema *Electronic Voting from Lattices with Verification* (EVOLVE), sendo uma sucessão do protocolo de Cramer et al. (1997). Como seu predecessor, ele possui as mesmas partes ativas, é construído para eleições de sim e não, podendo ser adaptado para 1 de n e conta com provas de conhecimento para validação de um voto e durante a etapa de contagem. No entanto, como no de Chillotti et al. (2016), todas as autoridades são necessárias para contagem dos votos.

O elemento central do protocolo é um esquema de comprometimento homomórfico para a operação de adição que depende dos problemas módulo-SIS e módulo-LWE. O esquema é uma adaptação de (BAUM et al., 2016) que originalmente foi construído para R-SIS e R-LWE em combinação com provas de conhecimento zero.

No entanto, não é incorporado no esquema ferramentas adequadas para verificabilidade de um voto. Isto é consequência do protocolo depender da honestidade de todas as autoridades e dispositivos de votação, característica que qualifica-se como um obstáculo da verificabilidade. Essa premissa existe especialmente como repercussão de uma autoridade maliciosa no esquema poder, sem ter que apresentar nenhuma evidência, bloquear cédulas construídas corretamente de serem consideradas no resultado final.

No artigo, também são apresentados os resultados de uma implementação protótipo do protocolo, sendo divulgados a parâmetros de configuração para o esquema de comprometimento, número de votantes e autoridades e é discutido como eles devem ser selecionados.

3.3 EPOQUE: PRACTICAL END-TO-END VERIFIABLE POST-QUANTUM-SECURE E-VOTING

Boyen et al. (2021) propõem o esquema Epoque com seu principal diferencial sendo a propriedade de E2E-V. Semelhante aos trabalhos supracitados, ele é capaz de instanciar eleições 1 de n , no entanto, sua concepção vem da motivação de resolver um problema que ocorre em (CRAMER et al., 1997; PINO et al., 2017), o de autoridades de contagem maliciosas que podem invalidar indevidamente votos, que é um obstáculo para E2E-V.

A solução apresentada foi por meio de alterações no núcleo do protocolo, substituindo o sistema de encriptação de chave pública por um Identity Based Scheme (IBE) genérico com a propriedade de resistência contra *chosen plaintext attacks*. Para o fim de garantir a segurança contra computadores quânticos, o esquema pode ser instanciado com primitivas criptográficas baseadas em reticulados que são:

- Um esquema de comprometimento aditivamente homomórfico, sendo utilizado o esquema baseado em reticulados proposto em (PINO et al., 2017).
- Um esquema de provas de conhecimento zero não-interativas baseado em (PINO et al., 2017) para criação de provas de validades dos votos que passaram pela fase de *commit*.
- Um esquema IBE, sendo sugerida uma variação do algoritmo baseado em reticulados ABB (AGRAWAL et al., 2010) com uma decifragem mestre mais rápida por meio de implementação de um atalho.

O artigo também implementa uma instanciação própria do Epoque, com parametrização do IBE para diferentes níveis de segurança. Complementarmente, é citado que os resultados de Pino et al. (2017) são também aplicáveis para este esquema já que a adição do IBE adiciona mudanças negligíveis.

3.4 LATTICE-BASED PROOF OF SHUFFLE AND APPLICATIONS TO ELECTRONIC VOTING

Aranha et al. (2021) propõem um esquema para votação baseado em reticulados que permite estruturas não-triviais de cédulas, permitindo múltiplas escolhas, ranqueamento de candidatos ou combinações com listas de partidos. A principal contribuição apresentada é uma prova de embaralhamento de valores conhecidos para esquema de comprometimento baseado em reticulados, que garante que uma sequência de comprometimentos foi gerada por uma sequência embaralhada de mensagens sem revelar o vínculo de uma única mensagem com seu comprometimento, garantindo privacidade do votante.

O fluxo de tarefas é muito semelhante ao apresentado na Subseção 2.5.1 com o seguinte protocolo descrito por Aranha et al. (2021) para submetimento de cédulas até a contagem e verificação:

1. O dispositivo do votante compromete-se com a cédula por meio de um esquema de comprometimento e cifra a abertura gerada pelo processo. O comprometimento que esconde o valor do voto e a abertura cifrada são enviados para um participante do protocolo chamado de urna (*ballot box*).
2. Quando a contagem inicia, a urna remove qualquer material que pode identificar o votante do texto cifrado e envia para o servidor de embaralhamento.
3. O servidor de embaralhamento decifra as *aberturas*, verifica os comprometimentos e retorna as cédulas. Ele utiliza a prova de conhecimento para o embaralhamento de valores conhecidos proposta pelo artigo para comprovar que as cédulas são consistentes com os comprometimentos.
4. Um ou mais auditores inspecionam a urna e o servidor de embaralhamento

Tabela 2 – Visão geral dos esquemas de votação eletrônica pós-quânticos

	CHILLOTTI et al.	PINO et al.	BOYEN et al.	ARANHA et al.
Tipo	homomórfico	homomórfico	homomórfico	<i>mix-net</i>
Verificabilidade	individual e universal	não	E2E-V	limitada
Implementação	não	privada	privada	pública
Estrutura	1 de n	1 de 2 ou 1 de n	1 de n	m de n, não-trivial

Fonte: autoria própria

Para garantir que o texto cifrado contenha uma abertura válida do comprometimento do votante, é empregado o esquema de encriptação verificável de Lyubashevsky e Neven (2017) em conjunto com os demais elementos do protocolo. Deste modo, garante-se segurança e praticabilidade. Contudo, não é realista a possibilidade de atingir tanto as propriedades de privacidade quanto verificabilidade universal, em função de não ser possível para a urna publicar as cédulas cifradas e provas de cédula porque isso torna o servidor de embaralhamento capaz de associar um votante ao seu voto, quebrando a privacidade. Assim, Aranha et al. (2021) resignam-se com formas limitadas de verificabilidade.

Adicionalmente, é abordada a preocupação de votações em dispositivos maliciosos solucionável com a implementação de códigos de retorno para detecção de fraudes (GJØSTEEN; LUND, 2016) em apenas partes da cédula, em função deles não serem capazes de lidar corretamente com a estrutura mais complexa de uma cédula proposta.

No artigo de Aranha et al. (2021) os autores prototipam e divulgam a implementação do esquema como prova de conceito, apresentando os parâmetros e desempenho, sendo o maior fator limitante o tamanho da prova de conhecimento.

3.5 VISÃO GERAL

Comparando os esquemas acima, Boyen et al. (2021) e Aranha et al. (2021) apresentam maior nível de maturidade, considerando o cumprimento, de forma aceitável, das propriedades de verificabilidade e privacidade. O primeiro é vantajoso por apresentar a forma mais desejada de verificabilidade (E2E-V), já o segundo permite representações avançadas de uma cédula. Assim, percebe-se que a abordagem melhor se adéqua as necessidades do Helios e adiciona um novo grau de flexibilidade.

O protocolo de Chillotti et al. (2016), por usar FHE com *bootstrapping* torna o protocolo inviável considerando a performance, além do quesito do esquema não ter sido implementado. Já a abordagem de Pino et al. (2017) peca no quesito de verificabilidade do votante para ser utilizável no mundo real.

Na Tabela 2, são comparadas algumas das propriedades de cada esquema. É também importante citar que todos os protocolos apresentam um nível aceitável no quesito privacidade. Adicionalmente, a Tabela 3 compara a performance de esquema de Pino et al. (2017) com o

de Aranha et al. (2021). Seja ε o número de autoridades e τ número de votantes, o principal ponto observável é o comportamento mais consistente do esquema de Aranha et al. (2021) considerando tamanhos variáveis para ε e τ . Os resultados de Pino et al. (2017) podem ser também considerados para Boyen et al. (2021) como anteriormente mencionado.

Tabela 3 – Comparação de performance dos esquemas

Comparação	Tamanho do Voto	Tempo do Votante	Tamanho da Prova	Tempo do Proveedor
ARANHA et al.	120 KB	209 ms	22τ KB	33τ ms
PINO et al.	20ε KB	9 ms	$18\varepsilon\tau$ KB	150τ ms

Fonte: adaptado de (ARANHA et al., 2021)

4 PROTOCOLO SELECIONADO

Por possuir maior maturidade entre todos os protocolos apresentados, notavelmente apresentando um grau aceitável de verificabilidade em conjunto com a possibilidade de cédulas estruturalmente complexas, o protocolo selecionado para integração ao Helios foi o de Aranha et al. (2021). Adicionalmente, também foi considerado a presença de uma implementação pública que serve como ponto de partida e permite a replicabilidade das avaliações de desempenho, que também provaram a maior eficiência do protocolo em relação com os outros trabalhos avaliados.

Assim, será realizada uma análise aprofundada do protocolo, inicialmente os principais parâmetros serão expostos, contextualizando os fatores fundamentais para a construção das primitivas criptográficas e provas de conhecimento que serão posteriormente abordadas. Em seguida, os principais algoritmos serão definidos, fundamentais para a especificação do passo-a-passo do protocolo.

4.1 PRELIMINARES

Sejam $p, r \in \mathbb{N}^+$, $N = 2^r$ e um anel $R = \mathbb{Z}[X]/\langle X^N + 1 \rangle$. A estrutura algébrica no qual as primitivas criptográficas dependem é o anel $R_p = R/\langle p \rangle$, interpretável como um anel de polinômios módulo $X^N + 1$ com coeficientes inteiros módulo p .

Para amostras aleatórias necessárias para as primitivas do esquema, são amostrados valores aleatoriamente da distribuição gaussiana discreta. Primeiramente, deve ser definida a distribuição normal contínua sobre \mathbb{R}^k centrada em $\mathbf{v} \in \mathbb{R}^k$ com desvio padrão σ que é dada pela equação:

$$\rho(\mathbf{x})_{\mathbf{v}, \sigma}^N = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{-\|\mathbf{x} - \mathbf{v}\|^2}{2\sigma^2}\right).$$

A distribuição gaussiana discreta é obtida pela normalização da distribuição contínua em R^k , denotada por:

$$\mathcal{N}_{\mathbf{v}, \sigma}^k(\mathbf{x}) = \frac{\rho_{\mathbf{v}, \sigma}^{kN}(\mathbf{x})}{\rho_{\sigma}^{kN}(R^k)} \text{ onde } \mathbf{x} \in R^k \text{ e } \rho_{\sigma}^{kN}(R^k) = \sum_{\mathbf{x} \in R^k} \rho_{\sigma}^{kN}(\mathbf{x}).$$

Quando $\sigma = 1$ ou $\mathbf{v} = 0$, os valores são omitidos na notação.

4.2 PARÂMETROS

Os parâmetros para construção das primitivas são apresentados na Tabela 4 para prover maior contexto antes do aprofundamento dos esquemas de comprometimento e encriptação. Notavelmente, os parâmetros das primitivas são dependentes entre si, em função de elas estarem intrinsecamente ligadas para garantir o funcionamento de certos processos. Tendo como exemplo os parâmetros k para comprometimentos e κ para encriptação, é fundamental eles compartilharem a mesma dimensão para possibilitar a cifragem das aberturas dos comprometimentos.

Tabela 4 – Parâmetros dos esquemas de comprometimento e encriptação

Parâmetro	Explicação	Restrições
N, δ	Grau do polinômio $X^N + 1$ em R	1 : N, δ devem ser potências de 2 2 : $N \geq \delta \geq 1$
p	Módulo dos comprometimentos	1 : p deve ser primo 2 : $p = 2\delta + 1 \pmod{4\delta}$
β_∞	Limite da norma- ∞ de certos elementos	$\beta_\infty < p^{1/\delta}/\sqrt{\delta}$
k	Comprimento da matriz de comprometimento em R_p	-
n	Altura da matriz de comprometimento em R_p	-
v	Norma- l_1 máxima de elementos em \mathcal{C}	-
σ_C	Desvio padrão das gaussianas discretas	$\sigma_C = 22 \cdot v \cdot \beta_\infty \cdot \sqrt{kN}$
\mathcal{C}	Espaço de desafios	$\mathcal{C} = \{c \in R_p \mid \ c\ _\infty = 1, \ c\ _1 = v\}$
$\bar{\mathcal{C}}$	Conjunto das diferenças entre os elementos em \mathcal{C} excluindo 0	$\bar{\mathcal{C}} = \{c - c' \mid c \neq c' \in \mathcal{C}\}$
D_{β_∞}	Conjunto de elementos de norma- ∞ não maiores que β_∞	$D_{\beta_\infty} = \{x \in R_p \mid \ x\ _\infty \leq \beta_\infty\}$
σ_E	Desvio padrão das gaussianas discretas	$\sigma_E = 11 \cdot v \cdot \sqrt{kN(3 + \beta_\infty)}$
κ	Dimensão do espaço de mensagens na encriptação	$\kappa = k$
ℓ	Dimensão da matriz de encriptação	$\ell = k - n$
η	dimensão do \mathbf{u} em $\mathbf{T}\mathbf{u} = \mathbf{u}$	$\eta = n + 1$
q	Módulo para encriptação	1 : q deve ser primo 2 : $q > 24\sigma_E^2$ 3 : $q > 2p(2l \cdot N^2 \cdot \beta_\infty^2 + N + 1)$ 4 : $q = 2\delta + 1 \pmod{4\delta}$
τ	Número total de votos	$(\tau^\delta + 1)/ R_p < 2^{-128}$

Fonte: adaptado de (ARANHA et al., 2021)

4.3 PRIMITIVAS

Como anteriormente comentado, as duas primitivas criptográficas fundamentais do protocolo são o esquema de comprometimento de Baum et al. (2016) aninhado com o esquema de cifragem de Lyubashevsky e Neven (2017) construídos sobre o anel R_p . Dessa forma, nesta seção, elas serão esquematizadas, apresentando a sequência de passos dos seus algoritmos e, para o esquema de comprometimento, também serão expostas as provas de conhecimento zero que podem construídas com base em suas propriedades particulares.

4.3.1 Esquema de Comprometimento

Baum et al. (2018) apresenta a construção de um esquema de comprometimento aditivamente homomórfico baseado nos problemas computacionais de módulo-SIS para garantir a propriedade de *hiding* e módulo-LWE para garantir *biding* que foram abordadas na Subseção 2.2.4. O principal diferencial deste esquema comparado com os seus predecessores baseados em reticulados é dispensar a necessidade de realizar múltiplas iterações de protocolos de conhecimento zero para correção em função delas acumularem um erro não negligenciável.

Protocolo 3. Esquema de Comprometimento de Baum et al. (2018)

O esquema é uma tripla $(\text{KeyGen}_C, \text{Com}, \text{Open})$ que segue as especificações apresentadas na Subsecção 2.2.4.

1. Geração de chave KeyGen_C :

Entrada: λ .

Saída: pk .

- 1) para um comprimento k e uma altura $n + 1$.
- 2) $\mathbf{B}_1 = \begin{bmatrix} \mathbf{I}_n & \mathbf{B}'_1 \end{bmatrix}$ onde $\mathbf{B}'_1 \xleftarrow{\$} \mathbb{R}_p^{n \times (k-n)}$.
- 3) $\mathbf{b}_2 = \begin{bmatrix} \mathbf{0}^n & 1 & \mathbf{b}'_2 \end{bmatrix}$ onde $(\mathbf{b}'_2)^\top \xleftarrow{\$} \mathbb{R}_p^{k-n-1}$.
- 4) a chave é definida como $pk := \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{b}_2 \end{bmatrix}$.

2. Fase de Confirmação Com:

Entrada: chave $pk = \mathbf{B}$, mensagem m .

Saída: comprometimento $\llbracket m \rrbracket$ e abertura d .

- 1) amostre um $\mathbf{r}_m \xleftarrow{\$} D_{\beta_\infty}^k$.
- 2) compute $\llbracket m \rrbracket$:

$$\mathbf{B} \cdot \mathbf{r}_m + \begin{bmatrix} \mathbf{0} \\ m \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix} = \llbracket m \rrbracket.$$

- 3) $d = (m; \mathbf{r}_m, 1)$.

3. Fase de Revelação Open:

Entrada: chave $pk = \mathbf{B}$, comprometimento $\llbracket m \rrbracket = \begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix}$, abertura $d = (m; \mathbf{r}_m, f)$ com $f \in \bar{\mathcal{C}}$.

Saída: valor booleano que representa se d é uma abertura válida para o comprometimento $\llbracket m \rrbracket$.

- 1) verifique se:

$$f \cdot \begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix} \stackrel{?}{=} \mathbf{B} \cdot \mathbf{r}_m + f \cdot \begin{bmatrix} \mathbf{0} \\ m \end{bmatrix}.$$

- 2) defina $\sigma_C = 11 \cdot \beta_\infty \cdot v \cdot \sqrt{kN}$.
- 3) verifique se $\|\mathbf{r}_m[i]\| \leq 4\sigma_C \sqrt{N}$ para cada $i \in [k]$.
- 4) se ambas as condições provarem-se corretas, retorne 1, senão 0.

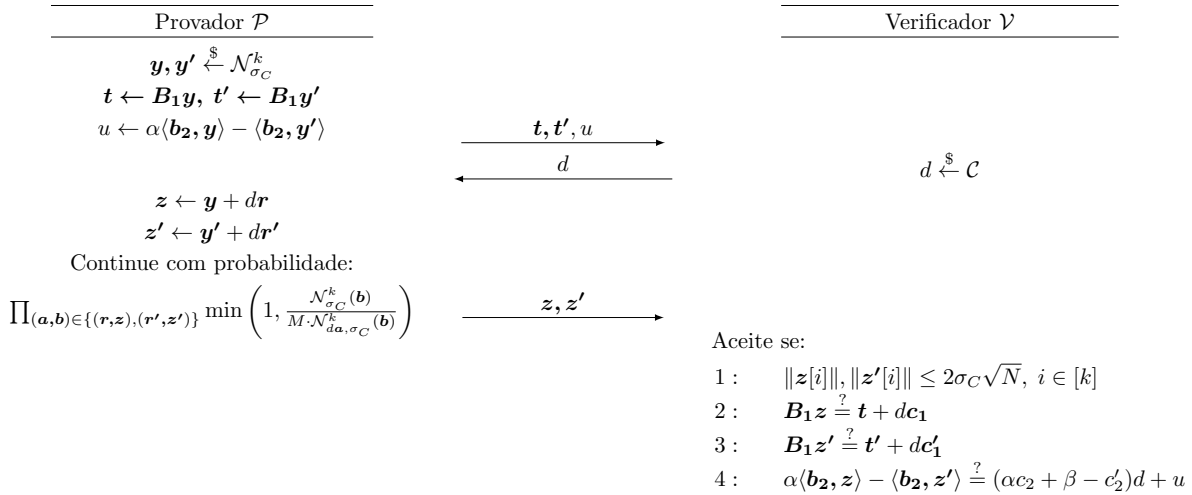
4.3.2 Provas de conhecimento

4.3.2.1 Relações lineares

Sejam $\llbracket x \rrbracket = \begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix}$ e $\llbracket x' \rrbracket = \begin{bmatrix} \mathbf{c}'_1 \\ c'_2 \end{bmatrix}$ dois comprometimentos construídos de acordo com o Protocolo 3 sobre uma mesma chave \mathbf{B} e com aberturas $(x; \mathbf{r}_x, f)$ e $(x'; \mathbf{r}'_{x'}, f')$ respectivamente. Baum et al. (2016) instanciam uma prova de conhecimento zero para provar a relação $x' = \alpha x$ entre as mensagens para algum $\alpha \in R_p$ público sem a divulgação dos valores de x, x', r, r' para o verificador.

Aranha et al. (2021) divulgam uma variação desta prova que envolve adicionalmente um $\beta \in R_p$ público, provando a relação $x' = \alpha x + \beta$. A instanciação deste protocolo é apresentada na Figura 8, sendo denotado como Π_{Lin} .

Figura 8 – Protocolo Π_{Lin}

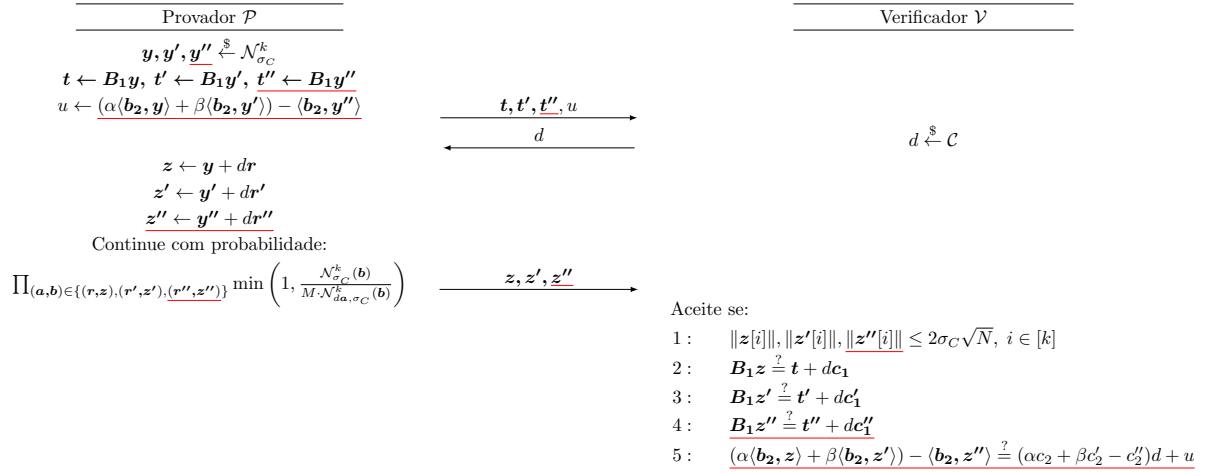


Fonte: Adaptado de (ARANHA et al., 2021)

4.3.2.2 Soma

Adicionalmente, o protocolo de votação demanda a prova de uma relação de soma para três comprometimentos $\llbracket x \rrbracket = \begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix}$, $\llbracket x' \rrbracket = \begin{bmatrix} \mathbf{c}'_1 \\ c'_2 \end{bmatrix}$ e $\llbracket x'' \rrbracket = \begin{bmatrix} \mathbf{c}''_1 \\ c''_2 \end{bmatrix}$, com aberturas $(x; \mathbf{r}_x, f)$, $(x'; \mathbf{r}'_{x'}, f')$ e $(x''; \mathbf{r}''_{x''}, f'')$ respectivamente, de forma que eles atendam a seguinte propriedade $x'' = \alpha x + \beta x'$ para $\alpha, \beta \in R_p$ públicos.

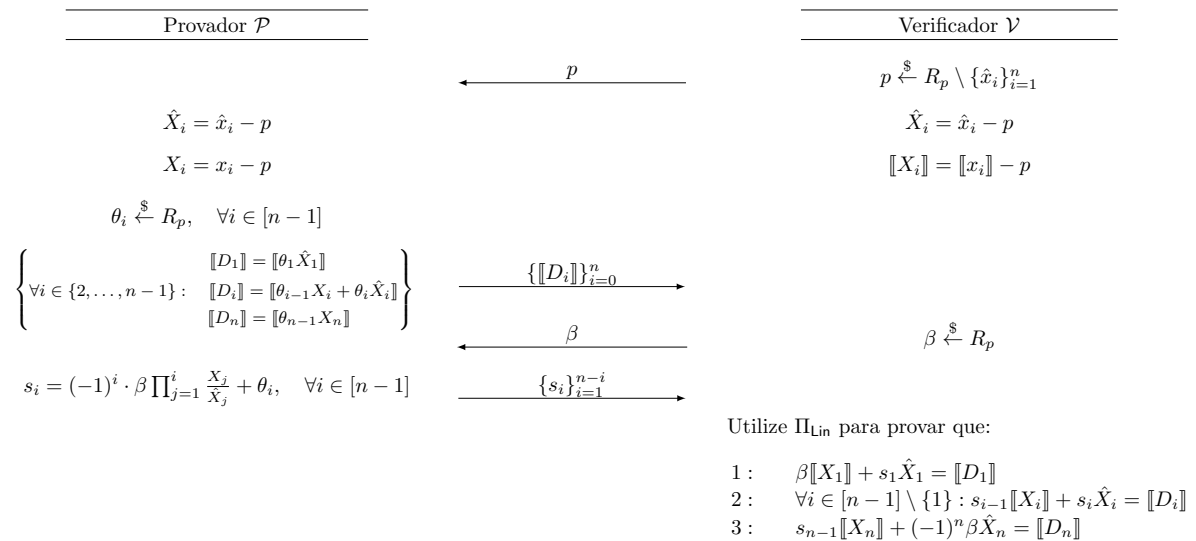
Baum et al. (2018) comenta a existência desta prova, mas não provê uma instância dela, apenas expressando sua semelhança com o protocolo acima. Desta forma, foi produzida uma variação de Π_{Lin} para atender a relação de soma, sendo denotada como Π_{Sum} .

Figura 9 – Protocolo Π_{Sum} , alterações realizadas pelo autor em relação ao Π_{Lin} estão marcadas em vermelho

Fonte: Adaptado de (ARANHA et al., 2021)

4.3.2.3 Embaralhamento de valores conhecidos

A principal contribuição de Aranha et al. (2021) é a prova de embaralhamento para valores conhecidos que opera como o mecanismo de anonimização dos votos para a *mix-net*. Sejam $\llbracket x_1 \rrbracket = \begin{bmatrix} c_{11} \\ c_{21} \end{bmatrix}, \dots, \llbracket x_n \rrbracket = \begin{bmatrix} c_{1n} \\ c_{2n} \end{bmatrix}$ uma sequência de comprometimentos $\llbracket \mathbf{X} \rrbracket$ para uma sequência de mensagens $\mathbf{X} = x_1, \dots, x_n$, com suas aberturas $(x_1; r_{x_1}, f_1), \dots, (x_n; r_{x_n}, f_n)$, uma permutação π em $\{1, \dots, n\}$ e uma sequência de mensagens embaralhadas $\hat{\mathbf{X}} = \hat{x}_1, \dots, \hat{x}_n$, de forma que para todo $i \in \{1, \dots, n\}$, $\hat{x}_{\pi(i)} = x_i$. Este protocolo é capaz de provar que $\llbracket \mathbf{X} \rrbracket$ são comprometimentos para $\hat{\mathbf{X}}$ de forma que não seja possível correlacionar um único comprometimento com a mensagem específica que o gerou.

Figura 10 – Protocolo Π_{Shuffle} Aceite se todas as instâncias de Π_{Lin} aceitarem

Fonte: Adaptado de (ARANHA et al., 2021)

4.3.3 Esquema de Encriptação Verificável

Aranha et al. (2021) apresentam uma variação do esquema Lyubashevsky e Neven (2017) que permite que qualquer um verifique que um texto cifrado possua certas propriedades. Neste caso, deseja-se provar que o texto plano é um valor $\boldsymbol{\mu} \in D_{\beta_\infty}^\kappa$ de forma que:

$$\mathbf{T}\boldsymbol{\mu} = \mathbf{u} \pmod{p},$$

para \mathbf{u} e \mathbf{T} fixos e $\mathbf{T} \in R_p^{\eta \times \kappa}$, a utilidade desta prova será futuramente abordada na Subsubseção 4.3.3.1. No entanto, é suficiente provar uma versão mais fraca desta asserção que mostra que a decifragem gera um $\bar{\boldsymbol{\mu}}$ e $\bar{c} \in \bar{\mathcal{C}}$ pequenos sobre R_p , de tal modo que:

$$\mathbf{T}\bar{\boldsymbol{\mu}} = \bar{c}\mathbf{u} \pmod{p}.$$

Protocolo 4. Esquema de encriptação verificável

O esquema é uma tupla de algoritmos ($\text{KeyGen}_V, \text{Enc}, \text{Ver}, \text{Dec}$). A abordagem realizada para tornar o esquema verificável acopla uma prova de conhecimento zero tornada não-interativa por Fiat-Shamir no texto cifrado.

1. Geração de chave KeyGen_V :

Entrada: λ .

Saída: pk, sk .

- 1) amostre $\mathbf{A} \xleftarrow{\$} R_q^{\ell \times \ell}$.
- 2) amostre $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} D_1^\ell$.
- 3) $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.
- 4) a chave pública é $pk = (\mathbf{A}, \mathbf{t})$.
- 5) a chave secreta é $sk = \mathbf{s}_1$.

2. Cifragem Enc:

Entrada: chave pública $pk = (\mathbf{A}, \mathbf{t})$, par (\mathbf{T}, \mathbf{u}) , mensagem $\boldsymbol{\mu} \in D_\beta^\kappa$, função de resumo criptográfico $H : \{0, 1\}^* \rightarrow \mathcal{C}$.

Restrições: $\mathbf{T}\boldsymbol{\mu} = \mathbf{u}$.

Saída: texto cifrado e .

- 1) amostre $\mathbf{r}, \mathbf{e} \xleftarrow{\$} D_1^{\ell\kappa}$.
- 2) amostre $\mathbf{e}' \xleftarrow{\$} D_1^\kappa$.
- 3)
$$\begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{A}^\top \otimes (p\mathbf{I}_\kappa) & p\mathbf{I}_{\ell\kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} \\ \mathbf{t}^\top \otimes (p\mathbf{I}_\kappa) & \mathbf{0}^{\kappa \times (\ell\kappa)} & p\mathbf{I}_\kappa & \mathbf{I}_\kappa \end{bmatrix} \begin{bmatrix} \mathbf{r} & \mathbf{e} & \mathbf{e}' & \boldsymbol{\mu} \end{bmatrix}^\top.$$

- 4) amostre $\mathbf{y} \leftarrow [\mathbf{y}_r \ \mathbf{y}_e \ \mathbf{y}_{e'} \ \mathbf{y}_\mu]^\top \xleftarrow{\$} D_{R^{(2\ell+2)\kappa}, \mathbf{0}, \sigma_E}$.
- 5) $\mathbf{Y} \leftarrow \begin{bmatrix} \mathbf{A}^\top \otimes (p\mathbf{I}_\kappa) & p\mathbf{I}_{\ell\kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} \\ \mathbf{t}^\top \otimes (p\mathbf{I}_\kappa) & \mathbf{0}^{\kappa \times (\ell\kappa)} & p\mathbf{I}_\kappa & \mathbf{I}_\kappa \\ \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times \kappa} & \mathbf{T} \end{bmatrix} \cdot \mathbf{y} \pmod{q}$.
- 6) $c \leftarrow H \left(\begin{bmatrix} \mathbf{A}^\top \otimes (p\mathbf{I}_\kappa) & p\mathbf{I}_{\ell\kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} \\ \mathbf{t}^\top \otimes (p\mathbf{I}_\kappa) & \mathbf{0}^{\kappa \times (\ell\kappa)} & p\mathbf{I}_\kappa & \mathbf{I}_\kappa \\ \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times \kappa} & \mathbf{T} \end{bmatrix}, \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \\ \mathbf{u} \end{bmatrix}, \mathbf{Y} \right)$.
- 7) $\mathbf{s} \leftarrow [\mathbf{r} \ \mathbf{e} \ \mathbf{e}' \ \boldsymbol{\mu}]^\top c$.
- 8) $\mathbf{z} \leftarrow \mathbf{s} + \mathbf{y}$.
- 9) com probabilidade $1 - \min \left(1, \frac{D_{R^{(2\ell+2)\kappa}, \mathbf{0}, \sigma_E}(\mathbf{z})}{3 \cdot D_{R^{(2\ell+2)\kappa}, \mathbf{s}, \sigma_E}(\mathbf{z})} \right)$, retorne ao passo 4.
- 10) se $\|\mathbf{z}\|_\infty \leq 6\sigma_E$, retorne ao passo 4, senão retorne $e = (\mathbf{v}, \mathbf{w}, c, \mathbf{z})$.

É possível observar a estrutura da prova de conhecimento zero não-interativa, mais especificamente, um protocolo- Σ (*com, ch, resp*) neste algoritmo que desempenha o papel de \mathcal{P} . Relembrando o Exemplo 2.2.1 e a simplificação de Fiat-Shamir apresentada, considerando como valores públicos a chave pública pk e o par (\mathbf{T}, \mathbf{u}) , podemos definir (\mathbf{v}, \mathbf{w}) como o comprometimento *com*, c como o desafio *ch* que foi substituído pela chamada a um oráculo (função H) e a resposta *resp* é \mathbf{z} .

3. Verificação Ver:

Entrada: texto cifrado $e = (\mathbf{v}, \mathbf{w}, c, \mathbf{z})$, par $x = (\mathbf{T}, \mathbf{u})$, chave pública $pk = (\mathbf{A}, \mathbf{t})$.

Saída: resultado da verificação de $\mathbf{T}\boldsymbol{\mu} = \mathbf{u}$ para a mensagem $\boldsymbol{\mu}$ que gerou a cifra e a partir da chave pk e do par x ;

- 1) se $\|\mathbf{z}\|_\infty > 6 \cdot \sigma_E$, retorne 0.
- 2) $\mathbf{Z} \leftarrow \begin{bmatrix} \mathbf{A}^\top \otimes (p\mathbf{I}_\kappa) & p\mathbf{I}_{\ell\kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} \\ \mathbf{t}^\top \otimes (p\mathbf{I}_\kappa) & \mathbf{0}^{\kappa \times (\ell\kappa)} & p\mathbf{I}_\kappa & \mathbf{I}_\kappa \\ \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times \kappa} & \mathbf{T} \end{bmatrix} - c \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \\ \mathbf{u} \end{bmatrix} \pmod{q}$.
- 3) se $c \neq H \left(\begin{bmatrix} \mathbf{A}^\top \otimes (p\mathbf{I}_\kappa) & p\mathbf{I}_{\ell\kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} & \mathbf{0}^{(\ell\kappa) \times \kappa} \\ \mathbf{t}^\top \otimes (p\mathbf{I}_\kappa) & \mathbf{0}^{\kappa \times (\ell\kappa)} & p\mathbf{I}_\kappa & \mathbf{I}_\kappa \\ \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times (\ell\kappa)} & \mathbf{0}^{\eta \times \kappa} & \mathbf{T} \end{bmatrix}, \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \\ \mathbf{u} \end{bmatrix}, \mathbf{Z} \right)$, retorne 0.
- 4) retorne 1.

Neste caso, é evidente a execução do papel de \mathcal{V} por este algoritmo. Notavelmente, o desafio c pode ser reconstruído por qualquer entidade que tenha conhecimento do texto cifrado e e dos valores públicos.

4. Decifragem Dec:

Entrada: chave secreta $sk = (s_1)$, par $x = (T, u)$, chave pública $pk = (A, t)$, texto cifrado $e = (v, w, c, z)$.

Saída: mensagem m, \bar{c} .

- 1) Se $\text{Ver}(e, x, pk) = 0$, retorne 0.
- 2) Repita até retornar a mensagem m :
 - 1) amostre $c' \xleftarrow{\$} C$
 - 2) $\bar{c} \leftarrow c - c'$
 - 3) $\bar{m}[i] \leftarrow (w - \langle s_1, v_1 \rangle) \bar{c} \pmod q$ para todo $i \in [\kappa]$
 - 4) Se $\|\bar{m}\|_\infty \leq q/2C$ e $\|\bar{m} \pmod p\|_\infty < 12\sigma_E$, retorne $(\bar{m} \pmod p, \bar{c})$.

4.3.3.1 Cifrando aberturas de comprometimentos

Na prática, o protocolo de votação cifra as aberturas dos comprometimentos. Uma propriedade essencial do Protocolo 3 para o funcionamento do esquema é a possibilidade de recuperar a mensagem m a partir de seu comprometimento $\llbracket m \rrbracket$ e a aleatoriedade r que o gerou. Assim, fundamentalmente, o protocolo de votação gera um comprometimento com o valor do voto e cifra a abertura. Dessa forma, o voto permanece oculto até o momento da contagem onde a abertura é decifrada.

Considerando a estrutura de um comprometimento

$$\llbracket m \rrbracket = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} B_1 \\ b_2 \end{bmatrix} \cdot r + \begin{bmatrix} 0 \\ m \end{bmatrix},$$

o polinômio c_1 é responsável por ligar $\llbracket m \rrbracket$ a um r único. Assim, a prova de relação do esquema de encriptação verificável $T\mu = u \pmod p$ para comprometimentos e suas aberturas cifradas deve mostrar que $B_1 r = c_1 \pmod p$.

4.3.4 Códigos de Retorno

Como mecanismo de proteção contra a possibilidade da votação ser realizada em dispositivos maliciosos, códigos de retorno são implementados primariamente por meio da chave k selecionada pelo sistema que instancia uma função pseudo-aleatória $\text{PRF}_k : \{0, 1\}^* \times R_p \rightarrow \{0, 1\}^n$ que transforma pares de sequências binárias de tamanho arbitrário e elementos de R_p em sequências binárias de tamanho n .

Durante o registro de um votante V , é aleatoriamente gerada uma chave secreta $a \xleftarrow{\$} R_p$ que tem a finalidade de ocultar o valor do voto. Considerando que os votantes possuem w opções de voto, definidas como $\hat{v}_1, \dots, \hat{v}_w \in R_p$, o pré-código \hat{r}_i para uma cédula \hat{v}_i é $\hat{r}_i = \hat{v}_i + a$ e o código de retorno é $r_i = \text{PRF}_k(V, \hat{r}_i)$.

É pré-computada para cada votante uma tabela de códigos de retorno para cada possibilidade de voto específica. Então, um votante quando instrui sua máquina qual sua escolha de voto ele também consulta a tabela obtendo o código de retorno esperado. No final do processo de votação, o votante com outro dispositivo recebe o código de retorno computado pelo servidor. Caso eles sejam distintos, é indicado que o voto selecionado pelo votante difere do recebido pelo servidor e ele é rejeitado.

Evidentemente, o servidor não pode descobrir o valor do voto para computar o código de retorno. Assim, como parte da cédula cifrada de um voto v_i , temos comprometimentos c_a , c_i e $c_{\hat{r}_i}$ para a chave do votante, o voto e o pré-código respectivamente. Dessa forma, é observável que é possível validar a integridade do pré-código $\hat{r}_i \stackrel{?}{=} v_i + a$ pela prova de soma (Figura 9) e é necessário apenas da abertura de $c_{\hat{r}_i}$ para computar e enviar o código de retorno ao usuário.

4.4 ALGORITMOS

Abaixo serão apresentados todos os algoritmos necessários para o esquema de votação de Aranha et al. (2021), o processo desconsidera os participantes como a urna e os auditores que serão abordados em conjunto com as fases do protocolo na seção subsequente.

Protocolo 5. Algoritmos de Votação de Aranha et al. (2021)

Os algoritmos são uma tupla (Setup, Register, Cast, Code, Count, Verify); as primitivas criptográficas necessárias são: um esquema de comprometimento ($\text{KeyGen}_C, \text{Com}, \text{Open}$) e um esquema de encriptação verificável ($\text{KeyGen}_{VE}, \text{Enc}_{VE}, \text{Ver}_{VE}, \text{Dec}_{VE}$).

1. **Configuração** Setup:

Entrada: fator de segurança λ .

Saída: chave pública pk , chave de decriptação dk , chave código ck .

- 1) $pk_C \leftarrow \text{KeyGen}_C$.
- 2) $(pk_V, dk_V) \leftarrow \text{KeyGen}_{VE}$.
- 3) $(pk_R, dk_R) \leftarrow \text{KeyGen}_{VE}$.
- 4) a chave pública é $pk = (pk_C, pk_V, pk_R)$.
- 5) a chave de decriptação é $dk = (pk_C, dk_V)$.
- 6) a chave de código é $ck = (pk_C, pk_V, dk_R)$.

2. **Registro** Register:

Entrada: chave pública $pk = (pk_C, pk_V, pk_R)$.

Saída: chave de verificação do votante vvk , chave de voto do votante vck , função de cédulas para pré-código f .

- 1) amostre $a \xleftarrow{\$} R_p$.

- 2) $(\mathbf{c}_a, d_a) \leftarrow \text{Com}(pk_C, a)$.
- 3) a chave de verificação do votante é $vvk = \mathbf{c}_a$.
- 4) a chave de voto do votante é $vck = (a, \mathbf{c}_a, d_a)$.
- 5) a função de cédulas para pré-códigos é $f(v) \rightsquigarrow v + a$.

3. Votar Cast:

Entrada: chave pública $pk = (pk_C, pk_V, pk_R)$, chave de voto do votante $vck = (a, \mathbf{c}_a, d_a)$, voto v .

Saída: cédula encriptada ev , prova da cédula Π^v .

- 1) $(\mathbf{c}, d) \leftarrow \text{Com}(pk_C, v)$.
- 2) $\hat{r} \leftarrow a + v$.
- 3) $(\mathbf{c}_{\hat{r}}, d_{\hat{r}}) \leftarrow \text{Com}(pk_C, \hat{r})$.
- 4) $\Pi_{\hat{r}}^{Sum}$ é a prova da soma que $\mathbf{c}_v, \mathbf{c}_a, \mathbf{c}_{\hat{r}}$ atendem a relação, $\hat{r} = v + a$ como especificado na Figura 9.
- 5) $(\mathbf{v}, \mathbf{w}, c, \mathbf{z}) \leftarrow \text{Enc}_{VE}(pk_V, d)$.
- 6) $\mathbf{e}_{\hat{r}} = (\mathbf{v}_{\hat{r}}, \mathbf{w}_{\hat{r}}, c_{\hat{r}}, \mathbf{z}_{\hat{r}}) \leftarrow \text{Enc}_{VE}(pk_R, d_{\hat{r}})$.
- 7) A cédula encriptada é $ev = (\mathbf{c}, d, \mathbf{v}, \mathbf{w}, c)$.
- 8) a prova da cédula é $\Pi^v = (\mathbf{z}, \mathbf{c}_{\hat{r}}, \mathbf{e}_{\hat{r}}, \Pi_{\hat{r}}^{Sum})$.

4. Código Code:

Entrada: chave de código $ck = (pk_C, pk_V, dk_R)$, chave de verificação do votante $vvk = \mathbf{c}_a$, cédula encriptada $ev = (\mathbf{c}, d, \mathbf{v}, \mathbf{w}, c)$, prova de cédula $\Pi^v = (\mathbf{z}, \mathbf{c}_{\hat{r}}, \mathbf{e}_{\hat{r}}, \Pi_{\hat{r}}^{Sum})$.

Saída: pré-código \hat{r} .

- 1) Verifique $\Pi_{\hat{r}}^{Sum}$.
- 2) Verifique $\mathbf{e} = (\mathbf{v}, \mathbf{w}, c, \mathbf{z})$ com $\text{Ver}_{VE}(pk_V, \mathbf{e})$.
- 3) Verifique $\mathbf{e}_{\hat{r}}$ com $\text{Ver}_{VE}(pk_R, \mathbf{e}_{\hat{r}})$.
- 4) Se qualquer verificação falhar, interrompa o algoritmo.
- 5) Decifre $d_{\hat{r}} \leftarrow \mathbf{e}_{\hat{r}}$.
- 6) Recupere o código \hat{r} a partir de seu comprometimento $\mathbf{c}_{\hat{r}}$ e abertura $d_{\hat{r}}$.

5. Contagem Count:

Entrada: chave de decriptação $dk = (pk_C, dk_V)$, cédulas encriptadas ev_1, \dots, ev_τ .

Saída: votos embaralhados v_1, \dots, v_τ , prova de embaralhamento Π^c .

- 1) Para cada cédula encriptada $ev_i = (\mathbf{c}_i, d_i, \mathbf{v}_i, \mathbf{w}_i, c_i)$, compute $d_i \leftarrow \text{Dec}_{VE}(dk_V, \mathbf{v}_i, \mathbf{w}_i, c_i)$.
- 2) Recupere a mensagem v'_i a partir do comprometimento \mathbf{c}_i e da abertura d_i .
- 3) Se qualquer decifragem falhar, interrompa o algoritmo.
- 4) Selecione uma permutação aleatória π em $\{1, 2, \dots, l\}$.
- 5) Atribua $v_{\pi(i)} = v'_i$. Assim, temos uma sequência v_1, \dots, v_τ embaralhada de votos.
- 6) Crie uma prova de embaralhamento Π^c para v_1, \dots, v_τ .

6. Verificação Verify:

Entrada: chave pública $pk = (pk_C, pk_V, pk_R)$, cédulas encriptadas ev_1, \dots, ev_τ , votos embaralhados v_1, \dots, v_τ , prova de embaralhamento Π^c .

Saída: valor booleano que representa se os comprometimentos das cédulas encriptadas foram gerados com os votos embaralhados.

- 1) Para cada cédula encriptada $ev_i = (\mathbf{c}_i, d_i, \mathbf{v}_i, \mathbf{w}_i, c_i)$, extraia \mathbf{c}_i , assim, obteremos uma sequência ordenada de comprometimentos $\mathbf{c}_1, \dots, \mathbf{c}_\tau$.
- 2) Verifique Π^c para os valores $\mathbf{c}_1, \dots, \mathbf{c}_\tau$ e v_1, \dots, v_τ se válido retorne 1, senão, 0.

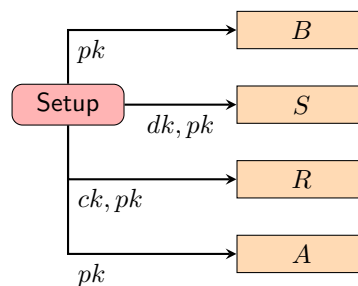
4.5 PROTOCOLO

O protocolo é uma sequência de fases (*setup, registration, casting, counting*). Temos como participantes uma urna eleitoral B , um embaralhador S , um gerador de códigos de retorno R , um conjunto de auditores A e um conjunto de votantes \mathbf{V} , cada um com seu computador D e celular F .

4.5.1 Fase de Configuração

Na fase configuração *setup*, apresentada pela Figura 11, um conjunto confiável de participantes executa o algoritmo Setup. A chave pública pk é distribuída para todos os participantes, a chave de deciptação dk é dada para S e a chave de código ck é dada para R .

Figura 11 – Fase de configuração

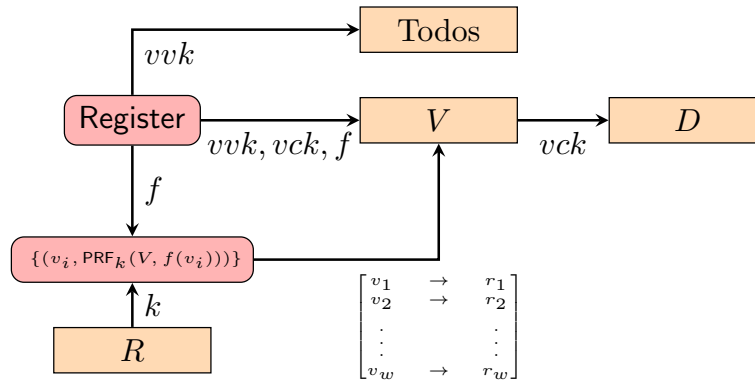


Fonte: autoria própria

4.5.2 Fase de Registro

Na fase de registro *registration*, apresentada pela Figura 12, um conjunto de participantes confiáveis executam o algoritmo Register para cada votante V em \mathbf{V} , obtendo (vvk, vck, f) . A chave de verificação do votante vvk é pública e a chave de voto do votante vck é dada para seu computador D . Em seguida, R seleciona uma chave k para PRF e um conjunto de partes confiáveis para cada votante V computam a tabela de códigos de retorno $\{(v_i, \text{PRF}_k(V, f(v_i)))\}$ para um conjunto relativamente pequeno de cédulas v_1, \dots, v_w e entregam-na ao votante.

Figura 12 – Fase de registro



Fonte: autoria própria

4.5.3 Fase de Votação

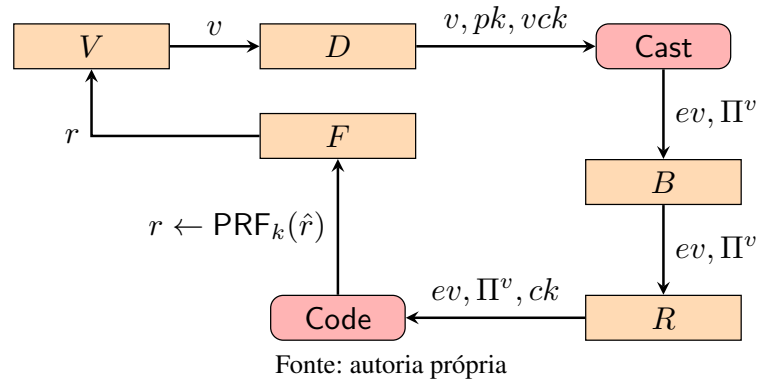
Na fase de votação *casting*, apresentada pela Figura 13, o votante V instrui seu computador D qual a cédula correta de votação. Em seguida, D executa o algoritmo Cast, enviando a cédula encriptada ev e a prova da cédula Π^v para a urna B que remete-as para o gerador de códigos de retorno R . Em seguida, R executa o algoritmo Code para obter o pre-código \hat{r} , computando o código de retorno $r \leftarrow \text{PRF}_k(\hat{r})$ que é enviado para o celular do votante F que remete para V . Então, V compara o código r com o código em sua tabela de códigos de retorno e apenas aceita a cédula como válida se eles são equivalentes.

Adicionalmente, o computador do votante D assina a cédula encriptada e a prova da cédula em nome do votante. A urna B e o gerador de códigos de retorno R verificam a assinatura, e, caso válida, R gera uma contra-assinatura de todo o conteúdo, enviando-a a D , que a verifica.

4.5.4 Fase de Contagem

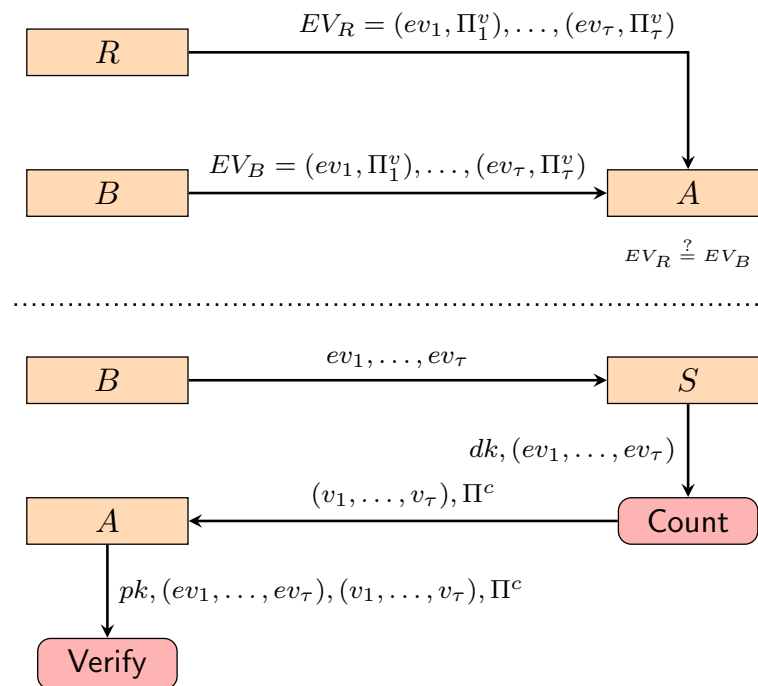
Na fase de contagem *counting*, apresentada pela Figura 14, a urna B e o gerador de códigos de retorno R enviam as cédulas encriptadas e provas da cédula para os auditores A . Se elas estão consistentes, os auditores aprovam. A urna B ordena as cédulas, considerando apenas os votos mais recentes de cada votante, e envia-as para o embaralhador S que usa o algoritmo

Figura 13 – Fase de votação



Count para revelar e embaralhar os votos v_1, \dots, v_l , computando a prova de embaralhamento Π^c ; ambos são enviados para os auditores, que utilizam o algoritmo Verify para verificar Π^c contra as cédulas encriptadas recebidas de B e R .

Figura 14 – Fase de contagem



5 HELIOS

Neste capítulo, será aprofundado o funcionamento do sistema Helios, com ênfase no processo de criação, votação e contagem de uma eleição. Brevemente, também será abordado o processo de autenticação, já que ele está intrinsecamente conectado ao registro de votantes. Adicionalmente, foi selecionada a variação desenvolvida pelo IFSC (CHAVES, 2023) por prover o modelo de autenticação LDAP que facilita o desenvolvimento e outras funcionalidades, incluindo melhorias na interface e mais possibilidades de operações de usuários administradores. No entanto, é importante ressaltar que o mecanismo criptográfico da aplicação baseado no protocolo de Cramer et al. (1997) com o esquema de chave pública ElGamal permanece inalterado em relação a sua versão oficial desenvolvida por Adida (2008).

Contextualizando a aplicação, o sistema Helios utilizado neste trabalho foi desenvolvido em Python na versão 3.6 utilizando o *framework* de desenvolvimento web Django. O sistema também depende de serviços adicionais para armazenamento de dados, execução de tarefas assincronamente e envio e recebimento de mensagens que são essenciais para o funcionamento da aplicação; temos como dependências:

- *nginx*: um servidor web;
- *celeryworker*: um enfileirador de tarefas responsável por executar processos como envio de email e contagem de votos de forma assíncrona;
- *celerybeat*: um enfileirador de tarefas responsável por executar processos em intervalos agendados, como limpeza dos resultados dos processos realizados pelo *celeryworker*;
- *redis*: um *message broker* para comunicação da aplicação central com os enfileiradores de tarefas;
- *db*: um banco de dados Postgres;

5.1 AUTENTICAÇÃO

O Helios apresenta um mecanismo generalizado e dinâmico para administração da autenticação de usuários. Podem ser definidos em variáveis de ambiente quais módulos de autenticação estão disponíveis e qual deve ser considerado o padrão, sendo implementada autenticação interna por usuário e senha ou externa por meio de terceiros como Google, Github, LDAP, etc. Neste trabalho, utilizou-se a autenticação por LDAP, sendo empregado um servidor online de testes publicamente disponível que consta com os usuários padrões¹:

- *ou=mathematicians,dc=example,dc=com*: riemann, gauss, euler e euclid
- *ou=scientists,dc=example,dc=com*: einstein, newton, galieleo e tesla

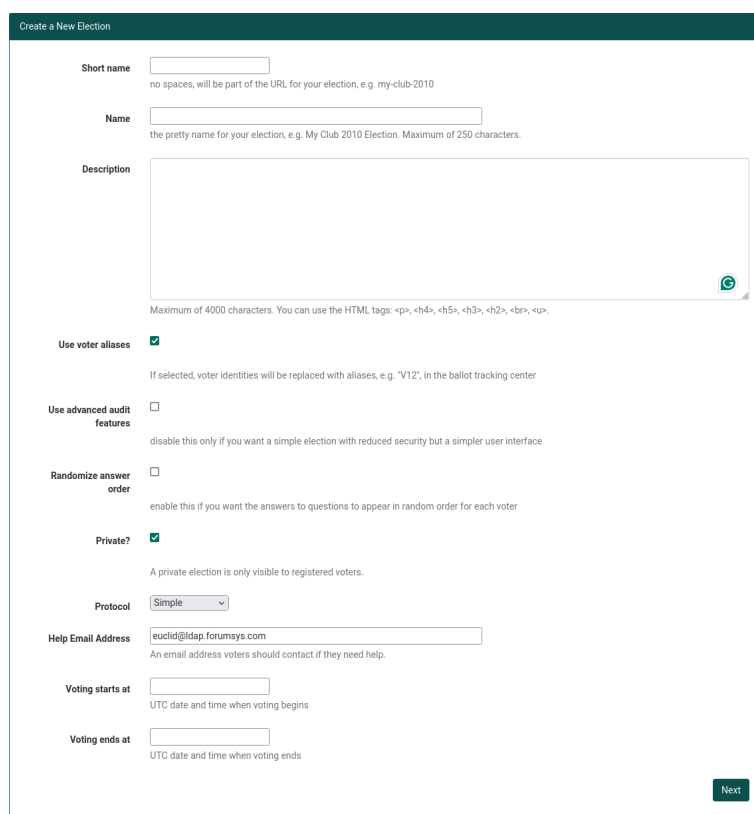
¹ Todos os usuários possuem a senha *password*

Uma funcionalidade provida pelo Django é uma interface exclusiva para usuários administradores que são manualmente cadastrados no ambiente de execução da aplicação. Destaca-se o mecanismo proporcionado para controle de permissões, de modo que um usuário do sistema demanda uma autorização prévia de um administrador para ser capaz de criar eleições.

5.2 CRIAÇÃO DE UMA ELEIÇÃO

Um usuário autorizado pode cadastrar eleições livremente no sistema. Inicialmente, são definidas as informações básicas pela interface apresentada na Figura 15, incluindo nome, descrição, início e fim da votação, entre outras, que podem posteriormente ser alteradas. Adicionalmente, quando uma eleição é criada, o sistema define o usuário como administrador e uma autoridade padrão para contagem do resultado, criando e armazenado um par de chaves ElGamal.

Figura 15 – Interface para cadastro de uma nova eleição



The screenshot shows a web form titled "Create a New Election" with a dark green header. The form contains the following fields and options:

- Short name:** A text input field with a note: "no spaces, will be part of the URL for your election, e.g. my-club-2010".
- Name:** A text input field with a note: "the pretty name for your election, e.g. My Club 2010 Election. Maximum of 250 characters."
- Description:** A large text area with a note: "Maximum of 4000 characters. You can use the HTML tags: <p>, <h4>, <h5>, <h3>, <h2>,
, <u>."
- Use voter aliases:** A checked checkbox with a note: "If selected, voter identities will be replaced with aliases, e.g. 'V12', in the ballot tracking center".
- Use advanced audit features:** An unchecked checkbox with a note: "disable this only if you want a simple election with reduced security but a simpler user interface".
- Randomize answer order:** An unchecked checkbox with a note: "enable this if you want the answers to questions to appear in random order for each voter".
- Private?:** A checked checkbox with a note: "A private election is only visible to registered voters."
- Protocol:** A dropdown menu currently set to "Simple".
- Help Email Address:** A text input field containing "euclid@ldap.forumsys.com" with a note: "An email address voters should contact if they need help."
- Voting starts at:** A text input field with a note: "UTC date and time when voting begins".
- Voting ends at:** A text input field with a note: "UTC date and time when voting ends".

A "Next" button is located at the bottom right of the form.

Fonte: Captura de tela realizada pelo autor

Cadastrada uma eleição, devem ser resolvidas pendências para que o processo de votação possa ser iniciado. É necessário o cadastro de questões e o registro de votantes, sendo opcional a configuração de autoridades, podendo ser utilizada a padrão criada no momento do cadastro. Após a resolução dos pré-requisitos, a eleição pode ser congelada, gerando a chave pública combinada das autoridades, inibindo qualquer alteração posterior e iniciando o processo de votação da data de início até a de fim.

5.2.1 Questões

Múltiplas questões podem ser cadastradas por eleição por meio da interface da Figura 16, com cada uma possuindo uma pergunta e múltiplas escolhas, um administrador adicionalmente define o número mínimo e máximo de opções selecionadas que constituem um voto válido e se a ordem das repostas será randomizada.

Figura 16 – Interface para cadastro de questões

Fonte: Captura de tela realizada pelo autor

Na lógica da aplicação, após cadastradas, questões são armazenadas como uma estrutura JSON. Considerando uma questão Q com n respostas a_1, a_2, \dots, a_n e endereços das respostas $url_1, url_2, \dots, url_n$, com um administrador definindo um máximo de escolhas i e um mínimo j , de forma que $j \leq i \leq n$. Temos a seguinte estrutura representada pela Figura 17.

Para o desenvolvimento do trabalho, os principais elementos a serem considerados são as respostas e o máximo e mínimo de escolhas, em função deles afetarem a estrutura dos votos diretamente e não como o resultado deve ser interpretado. Parâmetros como o tipo do resultado (*result_type*), podendo ser absoluto ou relativo, são para casos onde o número máximo de escolhas é igual a um, com o primeiro caso demandando que um candidato tenha $50\% + 1$ dos votos para ser eleito. Adicionalmente, o tipo de escolha (*choice_type*) e de contagem (*tally_type*) não apresentaram qualquer relevância na interpretação da questão. Dessa forma, assumiu-se que eles são elementos legados da aplicação.

5.2.2 Registro de votantes

Três opções são apresentadas para registro de votantes, apresentadas na Figura 18. A primeira, abre a eleição para todos os usuários autenticados no sistema, assim, o registro individual do votante é postergado para o momento do envio de sua primeira cédula. Dessa

Figura 17 – Estrutura de uma questão

```

1  {
2    "answer_urls": [url_1, url_2, ..., url_n]
3    "answers" : [a_1, a_2, ..., a_n],
4    "choice_type" : "approval",
5    "max": i,
6    "min": j,
7    "randomize_answer_order": False,
8    "result_type": "absolute",
9    "short_name": Q,
10   "tally_type": "homomorphic"
11 }

```

Fonte: autoria própria

forma, não são criadas entidades de votantes para usuários que não participaram da eleição. Já a segunda opção permite o envio de uma lista de votantes autorizados, podendo ser cadastradas credenciais exclusivas para a eleição ou pré-existentes em um dos sistemas de autenticação de terceiros. Neste caso, o registro de votantes ocorre no envio da lista. Por fim, a terceira tem o mesmo comportamento da segunda opção, com a limitação adicional de apenas aceitar usuários que utilizam o mesmo sistema de autenticação que o administrador da eleição.

Figura 18 – Interface para registro de votantes

Voters and Ballot Tracking Center

You can change this setting:

- anyone can vote ([click here for more details](#))
- only voters listed explicitly below can vote ([click here for more details](#))
- only voters from the same system ([click here for more details](#))

! You have to click this button to save your choice

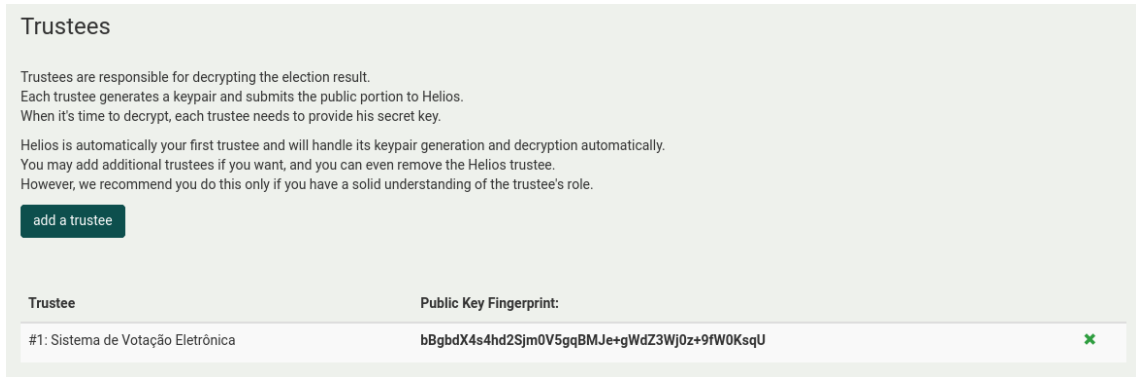
Fonte: Captura de tela realizada pelo autor

5.2.3 Configuração de autoridades

Um administrador da eleição pode configurar autoridades para computação do resultado, de forma que cada uma é responsável por gerar um par de chaves e enviar sua porção pública para construção da chave pública única da eleição. Como anteriormente mencionado, Helios por padrão cria uma autoridade que automatiza esse processo e a etapa final de contagem, no entanto, o administrador tem liberdade para adicionar e remover qualquer autoridade antes da eleição ser congelada. Além disso, diferente da autoridade padrão criada pelo Helios,

demais autoridades não armazenam sua chave privada na aplicação, sendo plenamente responsáveis para seguramente guardá-la até o momento da contagem onde os votos são decifrados.

Figura 19 – Interface para configuração de autoridades

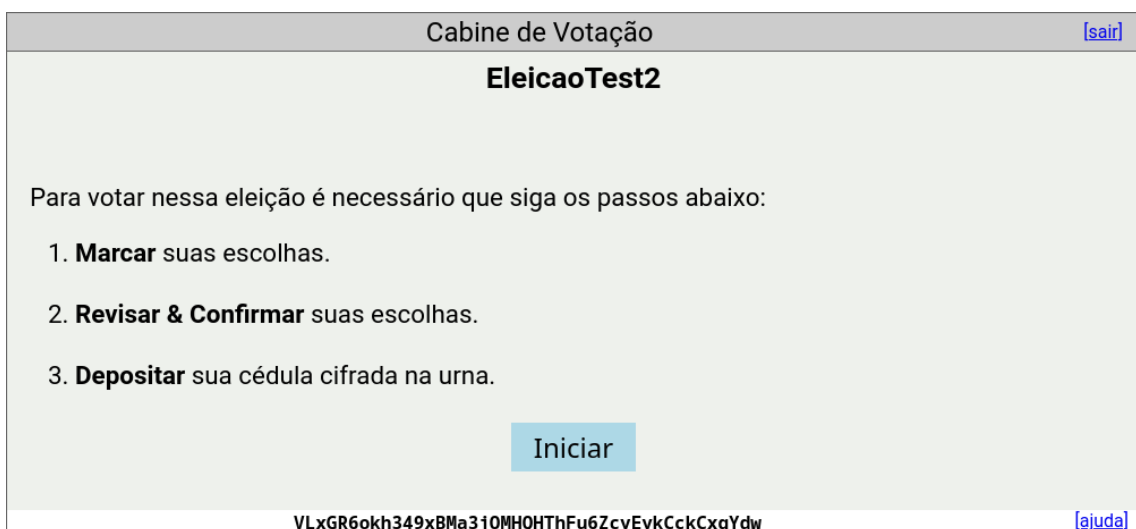


Fonte: Captura de tela realizada pelo autor

5.3 VOTAÇÃO

Durante a etapa de votação, um usuário autorizado é colocado em uma cabine eleitoral apresentada pela Figura 20. Abstraindo essa etapa ao nível de um usuário, deverão ser executadas três tarefas, que podem ser abortadas a qualquer momento, para registro de um voto, que são: 1) marcar as escolhas das questões (Figura 21); 2) revisar e confirmar as escolhas (Figura 22); e 3) depositar a cédula (Figura 23).²³

Figura 20 – Cabine eleitoral



Fonte: Captura de tela realizada pelo autor

Em mais detalhes, a primeira etapa apresenta todas as questões e suas repostas para o usuário, permitindo-o responde-las de acordo com os limites impostos previamente pelo número mínimo e máximo de escolhas. Seguindo para o próximo passo, a cabine eleitoral além de

apresentar para o votante suas escolhas, cifra a cédula de acordo com as decisões do votante. É importante mencionar que o processo criptográfico inteiro é executado no computador do votante de forma que o servidor não tenha acesso aos valores do voto em texto plano. Por fim, após a revisão, a cédula é depositada, enviando como conteúdo elementos identificadores da eleição e votos cifrados para cada questão.

Figura 21 – Interface para responder uma questão

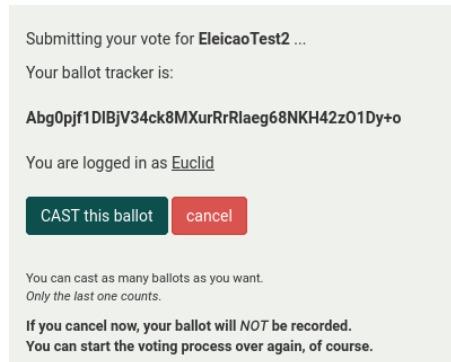
Fonte: Captura de tela realizada pelo autor

Figura 22 – Interface para revisão das repostas

Fonte: Captura de tela realizada pelo autor

Assim, para uma questão Q , com n respostas a_1, a_2, \dots, a_n , temos que cada resposta a_i carrega uma cifra ElGamal válida (a_i, b_{a_i}) e uma prova de validade da estrutura de um voto

Figura 23 – Interface para depósito da cédula



Fonte: Captura de tela realizada pelo autor

Figura 24 – Estrutura da cédula encriptada de uma questão

```

1  {
2    "choices": [
3      {"alpha": a_a_1, "beta": b_a_1},
4      ...,
5      {"alpha": a_a_n, "beta": b_a_n}
6    ],
7    "individual_proofs": [
8      {"challenge": c_a_1, "commitment": com_a_1, "response": resp_a_1},
9      ...,
10     {"challenge": c_a_n, "commitment": com_a_n, "response": resp_a_n}
11   ],
12   "overall_proof": [
13     {"challenge": "...", "commitment": "...", "response": "..."},
14     {"challenge": "...", "commitment": "...", "response": "..."}
15   ]
16 }

```

Fonte: autoria própria

que segue o protocolo- Σ apresentado na Figura 7, com um comprometimento com_{a_i} , desafio c_{a_i} e uma resposta $resp_{a_i}$. Adicionalmente, é embutida uma prova de validade geral para garantir que o número de escolhas está entre o mínimo e o máximo definidos. Novamente, a estrutura utilizada para armazenamento dos dados é um JSON, representada pela Figura 24.

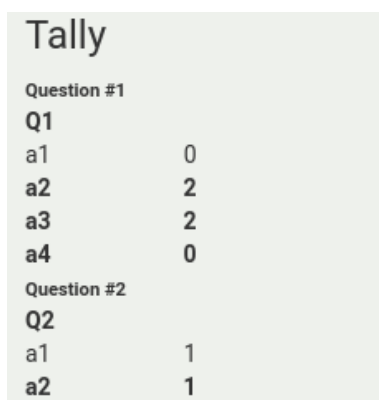
5.3.1 Verificação de um voto

No procedimento de verificação de uma cédula encriptada que ocorre após o envio do voto, são validados os elementos identificadores da eleição e, para cada questão, são verificadas as provas de conhecimento de suas escolhas e a prova geral que garante a integridade da estrutura do voto.

5.4 CONTAGEM

A qualquer momento durante a eleição ou após o período de votação, o administrador pode solicitar o processo de contagem de votos, que encerra a eleição. Nele, para cada resposta de cada questão, é realizada a adição homomórfica, apresentada na Proposição 2, de todos os votos. Por fim, as autoridades realizam a decifragem conjunta do produto final, obtendo o resultado das questões. Em seguida, o administrador pode divulgar o resultado da contagem para o público na forma da Figura 25.

Figura 25 – Exemplo de contagem



The image shows a screenshot of a 'Tally' screen. The title 'Tally' is at the top. Below it, 'Question #1' is followed by 'Q1'. There are four rows of options: 'a1' with 0 votes, 'a2' with 2 votes, 'a3' with 2 votes, and 'a4' with 0 votes. Below this, 'Question #2' is followed by 'Q2'. There are two rows of options: 'a1' with 1 vote and 'a2' with 1 vote.

Tally	
Question #1	
Q1	
a1	0
a2	2
a3	2
a4	0
Question #2	
Q2	
a1	1
a2	1

Fonte: Captura de tela realizada pelo autor

6 IMPLEMENTAÇÃO

Neste capítulo, será abordado o processo de implementação do protocolo de Aranha et al. (2021) no sistema Helios, com seu resultado disponível no Github¹. Como ponto de partida, foi utilizado o código-fonte que acompanha o trabalho de Aranha et al. (2021) que disponibiliza os esquemas de encriptação verificável (4.3.3) e comprometimento (4.3.1), além das provas de conhecimento para relações lineares (4.3.2.1) e embaralhamento de valores conhecidos (4.3.2.3).

O código foi desenvolvido na linguagem C, dependendo da biblioteca de teoria numérica FLINT (TEAM, 2023a) e da biblioteca para aritmética GMP (TEAM, 2023b). Assim, a primeira consideração na implementação do protocolo é a integração multi-linguagem do sistema Helios em Python e a implementação de Aranha et al. (2021) em C, levando em consideração a aplicação de modificações e adições para possibilitar ou facilitar o processo.

6.1 BIBLIOTECA COMPARTILHADA

A solução para este primeiro obstáculo foi a compilação do código de Aranha et al. (2021) como uma biblioteca compartilhada (*shared library*). Assim, é possível que um ambiente de execução em Python invoque facilmente as suas funções públicas por meio da biblioteca *ctypes*. Todavia, também foi necessário mapear todas as estruturas de C para classes e manualmente definir os argumentos e retorno das funções antes que elas possam ser invocadas. Por exemplo, a Figura 26 apresenta o processo de mapeamento da estrutura que corresponde a um esquema de comprometimento para uma classe, representando seus parâmetros como tuplas identificáveis por nome e por tipo da variável.

¹ github.com/JoaoPedro2002/helios-server-pqc

Figura 26 – Mapeamento de estrutura do esquema de comprometimento para uma classe

```

1 typedef struct _commitment_scheme_s {
2     nmod_poly_t  cyclo_poly;
3     pcrt_poly_t  irred;
4     pcrt_poly_t  inv;
5 } commitment_scheme_s;
6
7
8 class CommitmentScheme( ctypes.Structure ):
9     _fields_ = [
10         ("irred", PCRT_POLY_TYPE),
11         ("inv", PCRT_POLY_TYPE),
12         ("cyclo_poly", NMOD_POLY_TYPE)
13     ]

```

Fonte: autoria própria

Tabela 5 – Valores dos parâmetros dos esquemas de comprometimento e encriptação

Parâmetro	Esquema de Comprometimento	Esquema de Encriptação
N	1024	1024
δ	2	2
p	$\approx 2^{32}$	$\approx 2^{32}$
β_∞	1	1
k	3	-
n	1	-
v	36	36
σ	$\sigma_C \approx 54000$	$\sigma_E \approx 54000$
k	-	3
ℓ	-	2
η	-	2
q	-	2^{56}

Fonte: adaptado de (ARANHA et al., 2021)

Algumas modificações também foram pertinentes para o código em C. Primeiramente, os polinômios constantes das primitivas criptográficas foram encapsulados em estruturas, tornando-os externamente acessíveis. Suplementarmente, foi adicionado um método de recuperação da mensagem a partir de um comprometimento e sua abertura e, baseando-se na implementação já presente da prova de conhecimento para relações lineares e as definições de Baum et al. (2016), a prova de soma foi elaborada conforme as especificações na Figura 9. Por fim, foram desenvolvidas funções de utilidade para manipulação de polinômios. Notavelmente, métodos para conversão do formato de polinômios utilizados no esquema de comprometimento para o formato válido no esquema de encriptação.

Para a parametrização das primitivas criptográficas, foram mantidas as escolhas de Aranha et al. (2021). Relembrando os parâmetros da Tabela 4, temos como valores selecionados os dados da Tabela 5. Assim, é possível aproveitar as estimativas apresentadas no artigo para essa implementação. Considerando uma eleição com τ votantes, um voto individual tem tamanho 240KB, incluindo tanto a cédula encriptada quanto sua prova, e a estimativa de tamanho para a prova de embaralhamento é $\approx 22\tau$ KB.

6.2 ORGANIZAÇÃO DO PACOTE

Com todas as primitivas e provas de conhecimento necessárias disponíveis na biblioteca compartilhada, tornou-se possível o desenvolvimento do protocolo de votação. Adotando a organização do Helios, que isola o mecanismo criptográfico no pacote **crypto**, foi criado o pacote **crypto_pqc** que contém exclusivamente os componentes necessários para a construção do protocolo, sendo organizado da seguinte maneira:

- **compile.py**: carrega a biblioteca compartilhada e define os parâmetros constantes;

- **classes.py**: mapeia todas as estruturas para classes e define os argumentos e retornos das funções da biblioteca compartilhada;
- **commitment_scheme.py**: interface para invocação dos métodos referentes ao esquema de comprometimento de Baum et al. (2016);
- **encryption_scheme.py**: interface para invocação dos métodos referentes ao esquema de encriptação de Lyubashevsky e Neven (2017);
- **vericrypt.py**: interface dependente do esquema de Lyubashevsky e Neven (2017) com métodos referentes ao esquema de encriptação verificável;
- **primitives.py**: aglomera as primitivas criptográficas, permitindo a instanciação singular delas durante a execução da aplicação;
- **protocol_lin.py, protocol_sum.py, shuffle.py**: contém métodos para desempenhar tanto o papel do provador \mathcal{P} quanto de verificador \mathcal{V} para as prova de relações lineares, soma e embaralhamento de valores conhecidos respectivamente;
- **return_code_table.py**: contém todas as funções necessárias para computar a tabela de códigos de retorno, incluindo a função pseudo-aleatória;
- **scheme_algorithms.py**: apresenta a implementação de todos os algoritmos definidos no Protocolo 5;
- **players.py**: implementação de testes de todas as fases do protocolo de votação, incluindo testes de desempenho com quantidades variadas de votantes e diferentes números e estruturas de questões;
- **serializers.py**: mecanismo para serialização e desserialização das classes que representam estruturas para objetos JSON;
- **utils.py**: funções de utilidade, incluindo conversão de polinômios para *strings*, comparação de igualdade entre cédulas encriptadas e criação de questões e votos aleatórios.

Adicionalmente, para todas as primitivas e provas de conhecimento, foram desenvolvidos testes para garantir a integridade das operações, considerando especialmente a consistência dos dados que transitam para a biblioteca compartilhada. Aplicando os componentes em conjunto, foi implementada representações exemplo em mais alto nível, simulando eleições para os algoritmos e fases do protocolo.

6.2.1 Geração de números aleatórios

Para geração de números aleatórios essenciais para amostragem de polinômios ou obtenção de valores como a chave da função pseudo-aleatória, são empregados dois geradores. O

primeiro é uma implementação criptograficamente segura disponibilizada pela biblioteca criptográfica utilizada pelo Helios e, o segundo é específico da biblioteca FLINT para polinômios, tendo suas sementes geradas a partir do primeiro.

6.2.2 Estrutura de um voto

Um voto é representado por um polinômio v . Por exemplo, considerando uma questão Q com n opções a_1, \dots, a_n e o votante realiza duas escolhas arbitrárias a_i, a_j , onde $i > j$, então, temos um voto $0 \cdot x^n, \dots, 1 \cdot x^i, \dots, 1 \cdot x^j, \dots, 0 \cdot x^0$. Convertendo as escolhas de uma questão na estrutura de Figura 17 para um voto válido, basta selecionar a posição que os candidatos selecionados estão da ordem de escolhas e atribuí-las o coeficiente 1 no polinômio.

Notavelmente, a possibilidade de cédulas mais complexas dispensa a necessidade que sistema antigo do Helios apresenta de gerar uma prova de conhecimento por escolha da questão. Adicionalmente, para uma eleição com múltiplas questões, basta computá-las separadamente como se as fases de votação e contagem fossem realizadas diversas vezes. Evidentemente, uma tabela de códigos de retorno por questão deve ser gerada.

6.2.3 Função pseudo-aleatória

Como PRF, utilizou-se o algoritmo HMAC_SHA512. Assim, temos a chave e a imagem da função com 512 bits de comprimento. Praticamente, para possibilitar a aplicação da função em polinômios, primeiramente eles são convertidos em cadeias de caracteres, que são codificadas para bits. Para fins de padronização do trabalho com o atual sistema do Helios, a codificação utilizada sempre foi UTF-8.

6.2.4 Algoritmos

Seguindo as especificações do Protocolo 5 sem nenhuma alteração relevante, foi possível desenvolver os algoritmos com maior fidelidade ao artigo de Aranha et al. (2021). É observável por meio da Figura 27 e da Figura 28 a semelhança do resultado de código com as definições prévias do trabalho. Notavelmente, no segundo excerto de código, observam-se particularidades, como, por exemplo, a inicialização do polinômio a , que devem ser inseridas na implementação que no mais alto grau de abstração do Protocolo 5 não são especificadas.

Considerando a interoperabilidade entre o esquema de encriptação verificável e o de comprometimento, temos que o primeiro esquema utiliza polinômios no formato **fmpz** e o segundo no formato **nmod**. Assim, uma das considerações realizadas neste trabalho é a conversão entre esses formatos (Figura 29), que se mostra necessária nas etapas essenciais dos algoritmos para cifragem, verificação e decifragem das aberturas dos comprometimentos.

Figura 27 – Código para o algoritmo Setup

```

1 def setup():
2     # pk_C <- KeyGen_C
3     pk_C = commitment_scheme.keygen()
4     # (pk_V, dk_V) <- KeyGen_VE
5     pk_V, dk_V = encryption_scheme.keygen()
6     # (pk_R, dk_R) <- KeyGen_VE
7     pk_R, dk_R = encryption_scheme.keygen()
8     pk = (pk_C, pk_V, pk_R) # public key
9     dk = (pk_C, dk_V) # decryption key
10    ck = (pk_C, pk_V, dk_R) # code key
11    return pk, dk, ck
12

```

Fonte: autoria própria

Figura 28 – Código para o algoritmo Register

```

1 def register(pk):
2     pk_C, pk_V, pk_R = pk
3     # sample a <- R_p
4     a = NMOD_POLY_TYPE()
5     shared_library.nmod_poly_init(a, MODP)
6     shared_library.nmod_poly_randtest(a, flint_rand, DEGREE)
7     # (c_a, d_a) <- Com(pk_C, a)
8     c_a, d_a = commitment_scheme.commit(pk_C, a)
9     # the voter verification key is vvk = c_a
10    vvk = c_a
11    # the voter casting key is vck = (a, c_a, d_a)
12    vck = (a, c_a, d_a)
13    # the function f is v -> v + a
14    def _f(v):
15        return ballot_to_precode(shared_library, v, a)
16    f = _f
17
18    return vvk, vck, f
19

```

Fonte: autoria própria

6.2.5 Implementação de teste

Buscando seguir mais fielmente as fases do protocolo de Aranha et al. (2021) e dispensando as particularidades do Helios, foi desenvolvida uma implementação para testes que simula no código a comunicação entre participantes do protocolo. Em seu centro, temos a classe da Figura 30 que representa uma entidade capaz de enviar e receber mensagens que, no código, são parâmetros. Para manter o conceito de mensagens arbitrárias, fez-se o uso de elementos de reflexão.

Assim, mantendo o exemplo da configuração do protocolo, a Figura 31 segue fielmente as especificações da Subseção 4.5.1. Contudo, uma pendência na implementação é apresentada na fase de votação, onde não foi implementada a realização de assinaturas e contra-assinaturas

Figura 29 – Código para cifragem de aberturas dos comprometimentos

```

1 def encrypt_opening(pk, d, c: Commitment, pk_C: CommitmentKey) ->
  Veritext:
2   # convert c.c1 to FMPZ
3   u = c1_to_fmpz(shared_library, c.c1, commitment_scheme.scheme,
  vericrypt.context)
4   # convert pk_C.B1 to fmpz
5   t = b1_to_fmpz(shared_library, pk_C.B1, commitment_scheme.scheme,
  vericrypt.context)
6   # convert d to FMPZ_MOD_POLY_T * VECTOR
7   fmpz_d = opening_to_fmpz(shared_library, d, commitment_scheme.scheme,
  vericrypt.context)
8   # encrypt opening Ver_E
9   e, result = vericrypt.encrypt(t, u, fmpz_d, pk)
10  assert result, "Encryption failed for pk"
11  return e
12

```

Fonte: autoria própria

Figura 30 – Excerto da classe que representa um participante do protocolo

```

1 class Player:
2   def receive(self, **kwargs):
3       ...
4   def send_to_player(self, player, **kwargs):
5       ...
6   def add_to_params(self, **kwargs):
7       ...
8   def send_params_to_player(self, player, **kwargs):
9       ...
10  def send_to_all(self, **kwargs):
11      ...
12  def verify_value(self, value, key):
13      ...
14

```

Fonte: autoria própria

das cédulas encriptadas e suas provas por parte dos participantes.

Como apresentado na Subseção 4.5.3, na fase de votação, o computador do votante deve produzir uma assinatura da cédula encriptada e da prova de cédula que, caso válida, é contra-assinada pelo gerador de códigos de retorno. No entanto, este processo não foi implementado, decorrente das implementações populares de algoritmos de assinatura pós-quânticos em Python serem compatíveis com versões superiores a 3.6. Assim, como apresentado na Figura 32, a fase limita-se a execução dos algoritmos Cast e Code e o envio de cédulas encriptadas e provas para os participantes do protocolo.

Figura 31 – Implementação da fase de configuração

```

1 def setup_phase(self, questions: List[dict]):
2     self.questions = questions
3
4     # A trusted set of players run the setup algorithm Setup
5     pk, dk, ck = setup()
6     # The derived public key pk is given to every player
7     self._players["B"].send_to_all(pk=pk)
8     # The decryption key dk is given to the shuffler S
9     self._players["S"].receive(dk=dk)
10    # The code key ck is given to the return code generator R
11    self._players["R"].receive(ck=ck)
12

```

Fonte: autoria própria

Figura 32 – Excerto da implementação da fase de votação

```

1 # the voter V instructs the voter's computer D which ballot to cast
2 voter.cast(votes)
3 # The ballot box B sends the ev and the ballot proof to R
4 self.ballot_box.code(voter.id)
5

```

Fonte: autoria própria

6.2.6 Testes de desempenho

Juntamente com a implementação de testes que representa mais cruamente o mecanismo criptográfico de uma eleição, foi implementada no Helios a possibilidade de executar apenas seu protocolo criptográfico original, dispensando a aplicação web. Assim, tornou-se possível a comparação dos protocolos e um ambiente isolado.

6.2.7 Metodologia

Para obtenção de métricas relacionadas ao desempenho, tanto dos algoritmos quanto dos protocolos, foram realizados testes para os diversos procedimentos, extraindo a média de tempo apresentada em 25 vezes execuções em uma máquina de processador AMD Ryzen 7 5700X rodando em 3.4GHz. O principal parâmetro a ser considerado é a quantidade de votantes, contudo, a quantidade de escolhas da questão também é relevante.

6.2.7.1 Resultados

Na Tabela 6 é avaliado o tempo de execução dos algoritmos do Protocolo 5 para múltiplos votantes, com cada um submetendo um voto aleatoriamente amostrado em R_p . Considerando variações de tempo decorrentes principalmente do fator probabilístico de algumas operações do esquema, observa-se que o tempo dos algoritmos cresce de forma linear com o

número de votantes, assemelhando-se as expectativas de crescimento do tempo dos algoritmos apresentadas por Aranha et al. (2021). Também, nota-se que a operação Setup independe do número de votantes.

Tabela 6 – Desempenho dos algoritmos

Algoritmo	Votantes		
	25	50	100
Setup	0.0037s	-	-
Register	0.0385s	0.0790s	0.1533s
Cast	2.8647s	5.9205s	11.5414s
Code	1.2145s	2.4539s	4.7985s
Count	0.7713s	1.6233s	3.2001s
Verify	0.1415s	0.2868s	0.5631s

Fonte: autoria própria

Na Tabela 7, compara-se o protocolo antigo do Helios com o novo implementado. Nestes testes, foi considerada uma questão com cinco respostas, sendo pertinente destacar que a variação no número de respostas é irrelevante para o tempo de esquema de Aranha et al. (2021), todavia, no Helios original, elas implicam diretamente no número de provas de conhecimento criadas.

Para ambos os esquemas, a fase de configuração independe do número de votantes que participaram da eleição. Observa-se que o protocolo de Aranha et al. (2021) possui um maior desempenho que o Helios durante a fase de votação, no entanto, tem perdas significativas na fase de contagem. Neste conjunto de testes, a quantidade de votantes não foi suficientemente relevante para alterar o tempo de contagem do Helios que possui sua maior carga computacional durante a votação, já que esta fase também inclui a verificação do voto. O protocolo de Aranha et al. (2021) por contar com o mecanismo de códigos de retorno e necessitar da prova de embaralhamento, executa operações pesadas em ambas as fases. Também é importante mencionar que como a biblioteca compartilhada é utilizada, vários processos do novo protocolo são executados em código compilado, sendo naturalmente mais veloz que a implementação em Python puro do esquema original do Helios.

Tabela 7 – Comparação do desempenho entre protocolos

	Helios			Aranha et al. (2021)		
	25	50	100	25	50	100
Configuração	0.0054s	-	-	0.0039s	-	-
Registro	0.0002s	0.0004s	0.0008s	0.0551s	0.1374s	0.2308s
Votação	5.3263s	10.6226s	21.2814s	4.2165s	8.1156s	16.5790s
Contagem	0.0404s	0.0429s	0.0456s	0.9735s	1.8178s	3.8109s

Fonte: autoria própria

Figura 33 – Novas tabelas para o protocolo pós-quântico

Table Name	Column Name	Data Type
helios_auditor	uuid	varchar(50)
	public_key	text
	encrypted_votes_from_return_code_serv	text
	encrypted_votes_from_ballot_bc	text
	ballots	text
	proof	text
	election_id	integer
	id	integer
helios_ballotbox	uuid	varchar(50)
	encrypted_votes	text
	election_id	integer
	public_key	text
	t	text
helios_returncodeserver	uuid	varchar(50)
	encrypted_votes	text
	prf_key	bytea
	election_id	integer
	ck	text
helios_pqcvoter	uuid	varchar(50)
	expected_return_code	text
	return_code_table	text
	vck	text
	public_key	text
helios_shuffleserver	uuid	varchar(50)
	encrypted_ballot	text
	dk	text
	election_id	integer
	id	integer

Fonte: Captura de tela retirada pelo autor

6.3 INTEGRAÇÃO COM O SISTEMA HELIOS

Para integração com o sistema Helios, foi considerada uma eleição com a característica de ser aberta ao público e ter a ordem de respostas não-aleatorizada. Primeiramente, foi adicionada durante a criação de uma eleição a opção do administrador selecionar o protocolo de votação desejado. Adicionalmente, foram incluídas tabelas no banco de dados (Figura 33) para os participantes do protocolo, contendo os parâmetros necessários, como chaves cédulas e provas de conhecimento.

É importante mencionar que essa não é a abordagem ideal. Preferencialmente, cada autoridade deve estar em um servidor isolado, com suas chaves secretas inacessíveis pela aplicação. Contudo, esse procedimento elevaria imensamente a complexidade do produto final, demandando a implementação de um canal seguro de mensagens que devem possuir tamanho fixo para evitar vazamento de informações. Dessa forma, ele está fora do escopo deste trabalho.

Como forma de possibilitar o armazenamento das estruturas criptográficas do protocolo, foram implementados serializadores que convertem arbitrariamente as classes que encapsulam polinômios em objetos JSON, com o processo contrário também sendo possível. Relembrando a estrutura da classe da Figura 26, temos como resultado, o JSON da Figura 34.

Durante a elaboração da eleição, o usuário tem liberdade de modificar livremente o protocolo de votação eletrônica desejado até ela ser congelada, que impede qualquer alteração em seus dados. Com a seleção do esquema de Aranha et al. (2021), uma eleição, após congelada, ao invés de construir a chave pública com as porções das autoridades, realiza a fase de configuração do protocolo, gerando as chaves por meio do algoritmo Setup e criando os participantes, com exceção dos votantes.

Figura 34 – Serialização de uma classe

```

1  {
2      "irred": [
3          "...",
4          "...",
5      ],
6      "inv": [
7          "...",
8          "...",
9      ],
10     "cyclo_poly": "...",
11 }

```

Fonte: autoria própria

Como a eleição é publicamente disponível, o registro de votantes é delegado para a submissão do primeiro voto. Assim, no envio da cédula, se o votante não existe, são geradas as suas chaves e computada a tabela de códigos de retorno para cada possibilidade de voto em cada questão. Em seguida, as respostas do votante são convertidas para polinômios e o algoritmo Cast é invocado, produzindo as cédulas encriptadas e provas de cédula. Por fim, o votante armazena os códigos de retorno esperados.

Por meio das cédulas encriptadas e suas provas, o servidor de códigos de retorno obtém os códigos por meio do algoritmo Code e da PRF, enviando-os para o usuário que permite a cédula ser armazenada na urna caso eles sejam equivalentes aos códigos esperados. No entanto, esta etapa apresenta outra limitação da implementação, todas as operações comentadas são realizadas pelo servidor, sendo correto o votante cifrar a cédula e computar sua prova em seu próprio computador.

Considerando o protocolo antigo do Helios, construir a cédula encriptada no computador do votante é uma tarefa mais simples que é executada ao nível do cliente pelo navegador. No entanto, isto não é trivial para o caso atual em função da dependência do trabalho na biblioteca compartilhada. Adicionalmente, essa limitação remove o propósito de códigos de retorno já que todos os processos são realizados em um único dispositivo. Assim, uma possível abordagem seria o desenvolvimento de um módulo externo ao navegador para cifragem da cédula e um aplicativo utilizável em outro dispositivo para validação dos códigos de retorno.

Ao final de uma eleição, temos a principal diferença de comportamento de um protocolo de votação homomórfico para um baseado em *mix-nets*. Originalmente, o processo de contagem do Helios calcula a soma entre os votos, mas mantém o resultado cifrado até a reconstrução da chave secreta das autoridades. No novo protocolo, o resultado já é revelado, apenas aguardando o aval do auditor para ele ser publicado.

Isto conclui o novo processo de votação no Helios. De forma limitada, foi possível demonstrar a possibilidade de implementação de um novo protocolo na aplicação. Percebem-se

certas pendências no fluxo de tarefas que para tornar a contribuição prática em cenários reais devem ser corrigidas, notavelmente, o cálculo de cédulas de forma alguma deve ser realizado pelo servidor, no caso do Helios, ela é realizada pelo navegador do cliente por meio de código JavaScript, contudo esse é um processo extremamente complexo para implementação no novo protocolo. No entanto, a implementação mostra-se fiel a literatura em seus processos criptográficos. Adicionalmente não foram necessárias mudanças na interface, em função de ambos os protocolos poderem ser abstraídos para o fluxo padrão apresentado para esquemas de votação eletrônica.

7 CONCLUSÃO

Votação eletrônica é um tema delicado e complexo. Sua delicadeza vem do fato de ser um grande desafio manter níveis desejáveis de privacidade, integridade e verificabilidade em um meio rodeado de pontos de falha, onde erros nos resultados são inadmissíveis. Dessa forma, é difícil encontrar ou mesmo afirmar que exista um protocolo de votação ideal, sendo dever de uma aplicação definir quais desses pontos de falha são riscos aceitáveis. O sistema Helios, por exemplo, torna o processo de votação simples e acessível, no entanto, não existem garantias que um votante não foi coagido, ou que seu dispositivo não seja malicioso. Todavia, ambos problemas mencionados tem um risco baixíssimo nos seus casos de uso da aplicação.

A complexidade de votação eletrônica vem de ela ser um processo final dependente de diversas peças construídas em décadas de avanços criptográficos, com qualquer unidade falha podendo invalidar o sistema como um todo. Classicamente, o modelo de Cramer et al. (1997) prova-se seguro até hoje, mas diversas de suas garantias criptográficas são quebradas quando um adversário quântico é considerado. Assim, é sempre pertinente a busca contínua por desenvolvimentos criptográficos que consideram adversários futuros.

No caso deste trabalho, ele atingiu o objetivo de prover uma instanciação prática de um dos protocolos de votação pós-quânticos presentes na literatura, que, no atual momento, estavam apenas apresentados em forma descritiva. Adicionalmente, a monografia também tem valor como uma revisão do estado da arte de votação eletrônica, provendo fundamentação teórica para os esquemas clássicos e os resistentes a computadores quânticos, mostrando o que funciona atualmente, em conjunto com os aspectos que podem ser utilizados no futuro. Suplementarmente, foi apresentada, até o limite do conhecimento do autor, a primeira representação gráfica da prova de soma do esquema de comprometimento de Baum et al. (2018).

Comparando as opções de protocolos apresentadas, o esquema escolhido, mostrou-se distinto por basear-se em uma *mix-net*, mesmo que para um votante, o processo seja transparente independentemente do método de anonimização e contagem de votos, foi possível expor neste trabalho soluções que demandam a utilização de diversas primitivas criptográficas, mostrando como elas trabalham em conjunto para formar um sistema sólido e único. Além disso, fica evidente o valor de provas de conhecimento zero, permitindo com uma quantidade limitada de informação a formulação de elegantes garantias sobre asserções.

Considerando o processo de contagem de votos, é também possível afirmar que a mudança de um sistema homomórfico para uma *mix-net* que permite votos serem polinômios arbitrários no anel R_p abriu portas para a aplicação permitir eleições mais complexas. Muito discute-se sobre modelos de votação que melhor representam uma população como ranqueamento e voto único transferível (ERS, 2017) que agora tornam-se possíveis de serem implementados com no Helios.

Em suma, este trabalho considerou Helios como o caso de uso, contudo foi apresentada uma implementação genérica, com potencial para ser integrado em outros sistemas. Desconsiderando requisitos de infraestrutura, é relativamente simples a integração deste protocolo em

outras aplicações por meio dos algoritmos desenvolvidos. Todavia, adverte-se contra o uso deste trabalho em qualquer caso no mundo real. Mesmo desconsiderando as pendências apresentadas, tanto o produto de código produzido, quanto a biblioteca utilizada como dependência são provas de conceitos com apenas fins acadêmicos. Idealmente, uma equipe especializada deve sempre ser responsável por auditar a segurança da implementação de um protocolo de votação.

7.1 TRABALHOS FUTUROS

Como comentado, existem obstáculos que suas resoluções adicionariam um nível de complexidade inadequado para o escopo do trabalho. Assim, sugere-se como trabalhos futuros: 1) a separação dos participantes da eleição em servidores distintos; 2) a implementação de um módulo ou outra alternativa similar que permita a cifragem da cédula no lado do cliente, minimizando o aumento de complexidade da tarefa do votante; 3) o desenvolvimento de um aplicativo separado para verificação correta de códigos de retorno; e 4) a elaboração da funcionalidade de assinatura digital entre votantes com a urna e o servidor de códigos de retorno, que recentemente tornou-se possível com a atualização da versão oficial do Helios por Adida (2008) para Python 3.9.

REFERÊNCIAS

- ACAR, A. et al. A survey on homomorphic encryption schemes: Theory and implementation. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 51, n. 4, jul 2018. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3214303>.
- ADIDA, B. Helios: Web-based open-audit voting. In: **USENIX security symposium**. [S.l.: s.n.], 2008. v. 17, p. 335–348.
- ADIDA, B. et al. Electing a university president using open-audit voting: Analysis of real-world use of helios. **EVT/WOTE**, v. 9, n. 10, 2009.
- ADIDA, B.; MARNEFFE, O. de; PEREIRA, O. **Helios Election System**. 2023. Online: <https://github.com/benadida/helios-server>.
- AGRAWAL, S.; BONEH, D.; BOYEN, X. Efficient lattice (h) ibe in the standard model. In: SPRINGER. **Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29**. [S.l.], 2010. p. 553–572.
- AJTAI, M. Generating hard instances of lattice problems. In: **Proceedings of the twenty-eighth annual ACM symposium on Theory of computing**. [S.l.: s.n.], 1996. p. 99–108.
- ALI, S. T.; MURRAY, J. An overview of end-to-end verifiable voting systems. **Real-World Electronic Voting**, Auerbach Publications, p. 189–234, 2016.
- ANARVOTE. **GitHub - anarvote/helios_lib: Helios Server (Helios is an end-to-end verifiable voting system) as library — github.com**. 2018. https://github.com/anarvote/helios_lib. [Accessed 16-11-2023].
- ARANHA, D. F. et al. Lattice-based proof of shuffle and applications to electronic voting. In: SPRINGER. **Cryptographers’ Track at the RSA Conference**. [S.l.], 2021. p. 227–251.
- BAUM, C. et al. More efficient commitments from structured lattice assumptions. In: SPRINGER. **International Conference on Security and Cryptography for Networks**. [S.l.], 2018. p. 368–385.
- BAUM, C. et al. Efficient commitments and zero-knowledge protocols from ring-sis with applications to lattice-based threshold cryptosystems. **IACR Cryptol. ePrint Arch.**, v. 2016, p. 997, 2016.
- BENALOH, J. Dense probabilistic encryption. In: **Proceedings of the workshop on selected areas of cryptography**. [S.l.: s.n.], 1994. p. 120–128.
- BENALOH, J. Ballot casting assurance via voter-initiated poll station auditing. **EVT**, v. 7, p. 14–14, 2007.
- BENALOH, J. et al. End-to-end verifiability. **arXiv preprint arXiv:1504.03778**, 2015.
- BENALOH, J.; VAUDENAY, S.; QUISQUATER, J.-J. **IACR 2010 Election Results — iacr.org**. 2010. <https://www.iacr.org/elections/2010/>. [Accessed 16-11-2023].

BENALOH, J. C.; YUNG, M. Distributing the power of a government to enhance the privacy of voters. In: **Proceedings of the fifth annual ACM symposium on Principles of distributed computing**. [S.l.: s.n.], 1986. p. 52–62.

BENALOH, J. D. C. **Verifiable Secret-Ballot Elections**. Tese (Doutorado) — Yale University, USA, 1987. AAI8809191.

BERNSTEIN, D. J. Cost analysis of hash collisions: Will quantum computers make shares obsolete. **SHARCS**, v. 9, p. 105, 2009.

BOS, J. et al. Crystals-kyber: a cca-secure module-lattice-based kem. In: IEEE. **2018 IEEE European Symposium on Security and Privacy (EuroS&P)**. [S.l.], 2018. p. 353–367.

BOYEN, X.; HAINES, T.; MÜLLER, J. Epoque: practical end-to-end verifiable post-quantum-secure e-voting. In: IEEE. **2021 IEEE European Symposium on Security and Privacy (EuroS&P)**. [S.l.], 2021. p. 272–291.

CHAUM, D.; PEDERSEN, T. P. Wallet databases with observers. In: SPRINGER. **Annual international cryptology conference**. [S.l.], 1992. p. 89–105.

CHAUM, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. **Communications of the ACM**, ACM New York, NY, USA, v. 24, n. 2, p. 84–90, 1981.

CHAVES, S. **GitHub - shirlei/helios-server: Helios server** — [github.com](https://github.com/shirlei/helios-server). 2023. <https://github.com/shirlei/helios-server>. [Accessed 16-11-2023].

CHEN, L. et al. **Report on Post-Quantum Cryptography**. [S.l.]: NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2016.

CHILLOTTI, I. et al. A homomorphic lwe based e-voting scheme. In: SPRINGER. **Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7**. [S.l.], 2016. p. 245–265.

CLARKE, D.; MARTENS, T. E-voting in estonia. **Real-World Electronic Voting: Design, Analysis and Deployment**, CRC Press Boca Raton, FL, p. 129–141, 2016.

CORTIER, V. et al. A generic construction for voting correctness at minimum cost-application to helios. **Cryptology ePrint Archive**, 2013.

CRAMER, R.; DAMGÅRD, I.; SCHOENMAKERS, B. Proofs of partial knowledge and simplified design of witness hiding protocols. In: SPRINGER. **Annual International Cryptology Conference**. [S.l.], 1994. p. 174–187.

CRAMER, R.; GENNARO, R.; SCHOENMAKERS, B. A secure and optimally efficient multi-authority election scheme. **European transactions on Telecommunications**, Wiley Online Library, v. 8, n. 5, p. 481–490, 1997.

CRANOR, L. F.; CYTRON, R. K. Design and implementation of a practical security-conscious electronic polling system. 1996.

DOWLIN, N. et al. Manual for using homomorphic encryption for bioinformatics. **Proceedings of the IEEE**, v. 105, n. 3, p. 552–567, 2017.

DUCAS, L. et al. Crystals-dilithium: A lattice-based digital signature scheme. **IACR Transactions on Cryptographic Hardware and Embedded Systems**, p. 238–268, 2018.

DUCAS, L.; MICCIANCIO, D. FHEW: Bootstrapping homomorphic encryption in less than a second. In: OSWALD, E.; FISCHLIN, M. (Ed.). **Advances in Cryptology – EUROCRYPT 2015**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. p. 617–640. ISBN 978-3-662-46800-5.

DWORKIN, M. et al. **Advanced Encryption Standard (AES)**. [S.l.]: Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2001.

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

ERS. **Single Transferable Vote — electoral-reform.org.uk**. 2017. <https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/single-transferable-vote/>. [Accessed 05-06-2024].

FIAT, A.; SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In: **Conference on the theory and application of cryptographic techniques**. [S.l.: s.n.], 1986. p. 186–194.

FORNEY, G. D. 6.451 principles of digital communication ii, spring 2003. 2003.

FOUQUE, P.-A. et al. Falcon: Fast-fourier lattice-based compact signatures over ntru. **Submission to the NIST’s post-quantum cryptography standardization process**, v. 36, n. 5, 2018.

GENTRY, C. Fully homomorphic encryption using ideal lattices. In: **Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing**. New York, NY, USA: Association for Computing Machinery, 2009. (STOC ’09), p. 169–178. ISBN 9781605585062. Disponível em: <https://doi.org/10.1145/1536414.1536440>.

GJØSTEEN, K.; LUND, A. S. An experiment on the security of the norwegian electronic voting protocol. **Annals of Telecommunications**, Springer, v. 71, p. 299–307, 2016.

GLONDU, S. **GitHub - glondu/belenios: Verifiable online voting system. This is a mirror of https://gitlab.inria.fr/belenios/belenios — github.com**. 2023. <https://github.com/glondu/belenios>. [Accessed 16-11-2023].

GOLDREICH, O. **Foundations of Cryptography: Volume 1**. USA: Cambridge University Press, 2006. ISBN 0521035368.

GOLDREICH, O. **Foundations of cryptography: volume 2, basic applications**. [S.l.]: Cambridge university press, 2009.

GOLDREICH, O.; GOLDWASSER, S.; MICALI, S. How to construct random functions. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 33, n. 4, p. 792–807, 1986.

GOLDREICH, O.; OREN, Y. Definitions and properties of zero-knowledge proof systems. **Journal of Cryptology**, Springer, v. 7, n. 1, p. 1–32, 1994.

GOLDWASSER, S.; MICALI, S.; RACKOFF, C. The knowledge complexity of interactive proof systems. **SIAM J. Comput.**, v. 18, n. 1, p. 186–208, 1989.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: **Proceedings of the twenty-eighth annual ACM symposium on Theory of computing**. [S.l.: s.n.], 1996. p. 212–219.

HALEVI, S.; MICALI, S. Practical and provably-secure commitment schemes from collision-free hashing. In: SPRINGER. **Annual International Cryptology Conference**. [S.l.], 1996. p. 201–215.

HAO, F. **Schnorr non-interactive zero-knowledge proof**. [S.l.], 2017.

HAO, F.; RYAN, P. Y. A. **Real-World Electronic Voting: Design, Analysis and Deployment**. 1st. ed. USA: Auerbach Publications, 2016. ISBN 1498714692.

HARVEY, D.; HOEVEN, J. V. D. Integer multiplication in time $o(n \log n)$. **Annals of Mathematics**, Department of Mathematics, Princeton University Princeton, New Jersey, USA, v. 193, n. 2, p. 563–617, 2021.

HOFFMAN, K. **Linear algebra**. [S.l.: s.n.], 1971.

IACR. **IACR Requirements for E-Voting Systems — iacr.org**. 2008. <https://www.iacr.org/elections/eVoting/requirements.html>. [Accessed 12-11-2023].

JEAN, J. **TikZ for Cryptographers**. 2016. <https://www.iacr.org/authors/tikz/>.

JR, C. M. Report of the national workshop on internet voting: issues and research agenda. In: **ACM International Conference Proceeding Series**. [S.l.: s.n.], 2000. v. 128, p. 1–59.

KATZ, J.; LINDELL, Y. **Introduction to modern cryptography**. [S.l.]: CRC press, 2020.

KELSEY, J.; CHANG, S.-j.; PERLNER, R. Sha-3 derived functions: cshake, kmac, tuplehash, and parallelhash. **NIST special publication**, v. 800, p. 185, 2016.

KUSHILEVITZ, E.; OSTROVSKY, R. Replication is not needed: single database, computationally-private information retrieval. In: **Proceedings 38th Annual Symposium on Foundations of Computer Science**. [S.l.: s.n.], 1997. p. 364–373.

LYUBASHEVSKY, V.; NEVEN, G. One-shot verifiable encryption from lattices. In: SPRINGER. **Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I 36**. [S.l.], 2017. p. 293–323.

LYUBASHEVSKY, V.; PEIKERT, C.; REGEV, O. On ideal lattices and learning with errors over rings. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 60, n. 6, p. 1–35, 2013.

MADISE, Ü.; MARTENS, T. E-voting in estonia 2005. the first practice of country-wide binding internet voting in the world. In: GESELLSCHAFT FÜR INFORMATIK EV. **Electronic Voting 2006–2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting**. CC. [S.l.], 2006.

MAGKOS, E.; KOTZANIKOLAOU, P.; DOULIGERIS, C. Towards secure online elections: Models, primitives and open issues. **EG**, v. 4, p. 249–268, 01 2007.

MICCIANCIO, D. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. **computational complexity**, Springer, v. 16, p. 365–411, 2007.

NEUMANN, P. G. Security criteria for electronic voting. **NCSC**, v. 93, p. 478–482, 1993.

NGUYEN, P. Q.; STERN, J. The two faces of lattices in cryptology. In: SPRINGER. **International Cryptography and Lattices Conference**. [S.l.], 2001. p. 146–180.

NIST. **Selected Algorithms 2022 - Post-Quantum Cryptography | CSRC | CSRC — csrc.nist.gov**. 2022. <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. [Accessed 15-May-2023].

PEDERSEN, T. P. A threshold cryptosystem without a trusted party. In: SPRINGER. **Advances in Cryptology—EUROCRYPT’91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10**. [S.l.], 1991. p. 522–526.

PEIKERT, C. et al. A decade of lattice cryptography. **Foundations and trends® in theoretical computer science**, Now Publishers, Inc., v. 10, n. 4, p. 283–424, 2016.

PEREIRA, O. Internet voting with helios. **Real-World Electronic Voting: Design, Analysis and Deployment**, CRC Press, v. 8604, 2016.

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: Association for Computing Machinery, 2017. (CCS ’17), p. 1565–1581. ISBN 9781450349468. Disponível em: <https://doi.org/10.1145/3133956.3134101>.

PRINCETON. **Princeton Elections — princeton.heliosvoting.org**. 2019. <https://princeton.heliosvoting.org/>. [Accessed 16-11-2023].

REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 56, n. 6, p. 1–40, 2009.

RIVEST, R. L. et al. On data banks and privacy homomorphisms. **Foundations of secure computation**, Citeseer, v. 4, n. 11, p. 169–180, 1978.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 2, p. 120–126, feb 1978. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/359340.359342>.

ROMAN, S.; AXLER, S.; GEHRING, F. **Advanced linear algebra**. [S.l.]: Springer, 2005. v. 3.

SAKO, K.; KILIAN, J. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In: SPRINGER. **Advances in Cryptology—EUROCRYPT’95: International Conference on the Theory and Application of Cryptographic Techniques Saint-Malo, France, May 21–25, 1995 Proceedings 14**. [S.l.], 1995. p. 393–403.

SCHNEIER, B. **Applied Cryptography**. 2nd. ed. [S.l.]: Wiley, 1996. ISBN 0471117099.

SHAMIR, A. How to share a secret. **Communications of the ACM**, ACm New York, NY, USA, v. 22, n. 11, p. 612–613, 1979.

SHAMOS, M. I. Electronic voting-evaluating the threat. In: **Computers, Freedom and Privacy**. [S.l.: s.n.], 1993. v. 93, n. 3.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. **SIAM Review**, v. 41, n. 2, p. 303–332, 1999. Disponível em: <https://doi.org/10.1137/S0036144598347011>.

SUN, X. et al. Private machine learning classification based on fully homomorphic encryption. **IEEE Transactions on Emerging Topics in Computing**, v. 8, n. 2, p. 352–364, 2020.

TEAM, T. F. **FLINT: Fast Library for Number Theory**. [S.l.], 2023. Version 3.0.0, <https://flintlib.org>.

TEAM the G. **GNU MP: The GNU Multiple Precision Arithmetic Library**. [S.l.], 2023. Version 6.3.0, <http://gmplib.org/>.

Universidade Federal de Pelotas. **SOBRE O SISTEMA HELIOS VOTING — wp.ufpel.edu.br**. 2018. <https://wp.ufpel.edu.br/votacao/sobre-o-sistema-helios-voting/>. [Accessed 16-11-2023].

WEISSTEIN, E. W. "gaussian function,"mathworld-a wolfram web resource. <http://mathworld.wolfram.com/GaussianFunction.html>, 2012.

8 APÊNDICE 1: ARTIGO DO TCC

Votação Eletrônica Pós-Quântica Aplicada no Helios Voting

João Pedro Cardoso Barbosa¹

¹Universidade Federal de Santa Catarina (UFSC)

joao.c.barbosa@grad.ufsc.br

Abstract. *Electronic voting introduces a high degree of complexity in the search for a balance between its very high privacy requirement and the need for verifiability, demanding properties that are difficult to achieve together. Widely used in the academic environment, the Helios voting system guarantees voter privacy through homomorphic counting of ElGamal ciphers. However, the advent of quantum computing breaks the security of certain classical algorithms, enabling the violation of voter's privacy and compromising other crucial aspects of the election. This paper will therefore carry out a practical implementation of a post-quantum electronic voting protocol using Helios as a use case.*

Resumo. *Votação eletrônica introduz um alto grau complexidade na busca pelo equilíbrio entre seu altíssimo requisito de privacidade com a necessidade da verificabilidade, demandando propriedades dificilmente alcançáveis em conjunto. Muito utilizado no ambiente acadêmico, o sistema Helios de votação garante a privacidade do votante por meio de contagem homomórfica de cifras ElGamal. Todavia, o advento da computação quântica quebra a segurança de certos algoritmos clássicos, possibilitando a violação da privacidade do votante e comprometendo outros aspectos cruciais da eleição. Dessa forma, este trabalho realizará uma implementação prática de um protocolo de votação eletrônica pós-quântico considerando o Helios como caso de uso.*

1. Introdução

O advento de computadores proporcionou grandes benefícios e comodidades para a população, a maior escalabilidade, confiabilidade e presença de sistemas na sociedade possibilita a realização das mais diversas tarefas de maneira remota do conforto de casa. Assim, é um interesse lógico também digitalizar o processo tradicional de votação que demanda tempo e esforço considerável de todas as partes envolvidas. Todavia, votação eletrônica é um tema extremamente delicado, sendo sucessível a ataques efetivos e de difícil detecção. Integralmente, votação eletrônica é um processo que envolve preocupações humanas, digitais e éticas, não existindo respostas definitivas para todos os problemas presentes em seu meio.

Essencialmente, um sistema de votação deve balancear conveniência, privacidade e verificabilidade, sendo ineficiente se qualquer uma dessas propriedades não estiver em um grau de maturidade aceitável. Esse objetivo é de extrema dificuldade, especialmente considerando que o aperfeiçoamento de uma propriedade pode antagonizar outra. Assim, votação eletrônica é um processo que demanda criptografia.

Helios [Adida 2008] é um esquema de votação eletrônica que possui boas garantias tanto de privacidade quanto verificabilidade em combinação com a conveniência

concedida pelo votação remota via sistema web. Todavia, sua segurança é comprometida quando é considerado um adversário quântico. Dessa forma, esse trabalho propõem-se selecionar um protocolo de votação eletrônica pós-quântica presente na literatura e aplicá-lo no Helios.

2. Votação Eletrônica

Historicamente, as eleições se traduzem como um mecanismo que avalia o poder de escolha de um determinado grupo e, com a ininterrupta digitalização do mundo, é lógico modernizá-las para melhor adequá-las as comodidades oferecidas pela tecnologia da mesma forma que diversos outros processos foram avançados, por exemplo, *e-banking* e *e-commerce*. No entanto, votação é uma tarefa distinta, de forma que a disponibilidade de dados é um aspecto severamente limitante em função de seus altíssimos requisitos de privacidade.

Ainda considerando os exemplos de *e-banking* e *e-commerce*, esses sistemas esperam a ocorrência de fraude e arcam, normalmente, financeiramente com as consequências. No entanto, fraude em votações é inaceitável. Problema que se exacerba com os diversos fatores, especialmente humanos, que devem ser considerados para votações, venda de votos e coação são alguns dos aspectos que tornam o processo tão delicado e dependente da confidencialidade. Assim, é imposto um desafio de balancear o sigilo de um voto, com garantias da honestidade da eleição.

No âmbito digital, [Chaum 1981] introduziu o uso de criptografia para eleições, apresentando primitivas criptográficas para segurança durante comunicações anônimas. Em geral, esquemas de votação eletrônica cifram cédulas para ocultar o valor dos votos. Notavelmente, o desafio fundamental desses protocolos é principalmente o cálculo da contagem sem quebrar a privacidade dos eleitores e, ao mesmo tempo, conseguir provar que todos os votos foram contados corretamente. Para resolver esse problema, existem três estruturas principais: 1) *contagem homomórfica*: todas as cédulas cifradas são somadas e depois decifradas, então, nenhum voto singular é revelado; 2) *assinatura cega*: as cédulas são anonimizadas antes de serem enviadas; e 3) *mix-nets*: as cifras são embaralhadas de uma maneira que não existe nenhuma correlação entre uma cédula cifrada e o voto que ela contém.

[Cranor and Cytron 1996] uniram diversas abordagens de sistemas eletrônicos de votação apresentados na literatura e identificou quatro propriedades centrais essenciais em quase todos os sistemas eleitorais:

- **Precisão**: um voto válido é inalterável, enquanto um voto inválido não pode ser considerado para a contagem.
- **Democracia**: apenas votos de entidades registradas são considerados.
- **Privacidade**: não é possível para qualquer entidade ligar um votante a sua cédula de votação e nenhum votante é capaz de provar em quem votou.
- **Verificabilidade**: um votante pode independentemente verificar se seu voto foi corretamente contado e se a contagem final é legítima.

Definição 2.1. (End-to-End Verifiable (E2E-V)) Um sistema de votação eletrônica é considerado E2E-V se ele provê técnicas para votantes individualmente verificarem aspectos cruciais do resultado da eleição, sem a necessidade dos votantes confiarem no *software*, *hardware*, oficiais da eleição, procedimento ou observadores [Benaloh et al. 2015].

3. Primitivas Criptográficas

A complexidade de votação eletrônica vem de ela ser um processo final dependente de diversas primitivas construídas em décadas de avanços criptográficos, com qualquer unidade falha podendo invalidar o sistema na totalidade. Assim, serão apresentadas nesta seção as principais primitivas para tanto o protocolo do Helios [Adida et al. 2023], quanto o protocolo pós-quântico selecionado [Aranha et al. 2021].

3.1. Cifragem de Chaves Públicas (PKE)

Criptografia de chaves públicas utiliza duas chaves diferentes, uma pública e outra privada, sendo computacionalmente difícil deduzir a chave privada a partir da pública. O algoritmo fundamental do sistema é o gerador de chaves.

Definição 3.1 (Cifragem de Chave Pública (PKE)). Um sistema de cifragem de chave pública é uma tripla (K, E, D) de algoritmos [Goldreich 2006]:

- Geração de chaves $K(\lambda)$: recebe o parâmetro de segurança λ e retorna (sk, pk) que correspondem a chave privada e a chave pública respectivamente.
- Cifragem $E(pk, M)$: o algoritmo recebe a chave pública pk e uma mensagem M e retorna o texto cifrado C .
- Decifragem $D(sk, C)$: o algoritmo recebe a chave privada sk e um texto cifrado C e retorna a mensagem M .

Essencialmente, o processo utiliza o par de chaves para cifrar e decifrar mensagens, sendo normalmente baseado em funções alçapão, encriptação é o caminho fácil que demanda a chave pública, abertamente distribuída entre todas as partes. A decríptação é a direção difícil, com o segredo para solucioná-la sendo a chave privada.

3.2. Homomorfismo (Hom)

Na criptografia, homomorfismo é uma propriedade que possibilita a realização de operações em textos cifrados sem decríptação preliminar. Tradicionalmente, o processo de encriptação garante confidencialidade a informações por meio da geração de textos cifrados alheios ao conteúdo original, impossibilitando operações sobre os dados cifrados. Dessa forma, aplicações devem decríptar a informação antes de operar sobre ela. Isso, evidentemente, introduz problemas de privacidade, especialmente quando dada informação deve transitar em canais inseguros ou ser acessada por terceiros.

Assim, a motivação de encriptação homomórfica nasceu como solução para aplicações que demandam tanto a privacidade quanto a necessidade de operar facilmente sobre os dados. Podemos observar essa propriedade aproveitada em aplicações de votação eletrônica [Adida et al. 2023]; recuperação de informações privadas [Kushilevitz and Ostrovsky 1997]; *Machine Learning* [Sun et al. 2020]; e dados de saúde [Dowlin et al. 2017].

Definição 3.2 (Encriptação Homomórfica (Hom)). Um esquema criptográfico é considerado homomórfico sobre uma operação $*$ se ele suporta determinada equação:

$$E(m_1) * E(m_2) = E(m_1 * m_2), \forall m_1, m_2 \in M,$$

onde E é a operação de encriptação e M o conjunto de todas as mensagens possíveis [Acar et al. 2018].

3.3. Prova de conhecimento zero (ZK)

Provas de conhecimento zero foram inicialmente introduzidas por [Goldwasser et al. 1989] e, desde então, são valiosas ferramentas criptográficas que provêm evidências convincentes de que determinada asserção é verdadeira, de forma que o protocolo é equivalente a uma parte confiável garantindo a validade da asserção [Goldreich 2006]. Esses sistemas podem ser definidos como a comunicação entre duas partes que possuem conhecimento sobre certas informações públicas: o provador \mathcal{P} que deseja revelar a validade de alguma afirmação predeterminada ao verificador \mathcal{V} sem produzir nenhum outro conhecimento além da corretude da proposição.

Duas principais propriedades devem ser garantidas em provas de conhecimento zero [Goldreich and Oren 1994], 1) *completeness*, se a proposição é verdadeira, um \mathcal{V} honesto será convencido por um \mathcal{P} honesto da validade da asserção; e 2) *soundness*, se a proposição é falsa, é computacionalmente inviável um \mathcal{P} desonesto convencer um \mathcal{V} honesto que ela é verdadeira.

3.4. Esquema de Comprometimento (ComSch)

Esquemas de comprometimento são primitivas criptográficas que permitem uma parte se comprometer com um valor e mantê-lo oculto. [Goldreich 2006] compara o esquema com um envelope selado. Ao colocar uma nota nesse envelope, uma parte compromete-se com o conteúdo da nota, mas ainda o mantém em segredo.

Definição 3.3 (Esquema de Comprometimento (ComSch)). Um esquema de comprometimento é um protocolo de 2 fases que envolve uma parte chamada de *remetente*, que pode comprometer-se com um valor e um *receptor*. As fases do protocolo são:

1. **Fase de Confirmação:** o remetente é vinculado a um valor único (o comprometimento), com a fase não produzindo nenhuma informação deste valor para o receptor;
2. **Fase de Revelação:** a valor é revelado pelo remetente e o receptor verifica sua autenticidade.

Tipicamente, para o valor ser revelado durante a segunda fase é necessária uma chave chamada de abertura (*opening*), enviada pelo remetente durante a primeira ou a segunda fase do protocolo [Halevi and Micali 1996]. Adicionalmente, para os dois casos, são normalmente implementadas provas de conhecimento zero para impor um comportamento honesto das partes envolvidas.

Duas principais propriedades devem ser garantidas em um esquema de comprometimento, 1) *hiding*, para que no fim da primeira fase, nenhuma informação do valor do remetente seja possivelmente vazada; e 2) *biding*, de forma que o exista apenas um único valor que o receptor na segunda fase aceite como legítimo.

3.5. Secret sharing (SeSh)

Secret sharing é um método para distribuir um segredo para um grupo de forma que deve existir um acordo entre uma parcela das partes para reconstruir esse segredo. Essa primitiva permite a descentralização da responsabilidade de decifrar mensagens, impedindo que uma entidade única seja capaz de decifrar individualmente as informações. É possível visualizar o fluxo pela Figure 1, nela temos uma chave secreta S responsável por revelar

a contagem de votos dividida entre n autoridades. Mesmo que algumas entidades sejam maliciosas e não desejem contribuir para revelação, por exemplo, a Autoridade n , ainda é possível obter o resultado desejado.

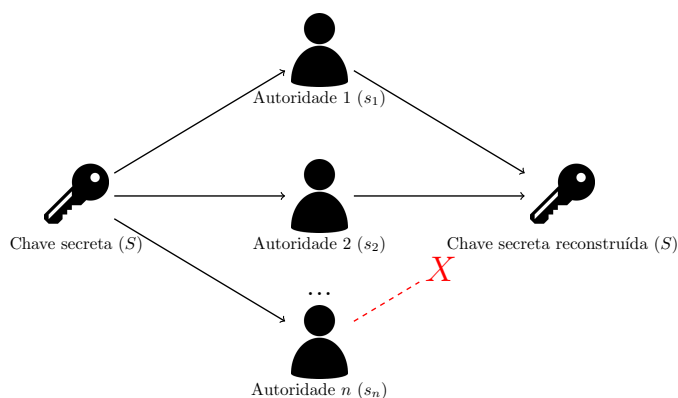


Figura 1. Sistema multi-autoridade

4. Helios

Helios [Adida 2008] é um sistema de votação eletrônica multi-autoridade E2E-V desenvolvido em Python que segue o modelo de contagem homomórfica de [Cramer et al. 1997], permitindo eleições n para m , ou seja, um número arbitrário de opções para um número arbitrário de respostas. Sua aplicação ideal são em cenários que priorizam integridade e privacidade, mas o risco de coerção é baixo [Ali and Murray 2016] em função de existirem pouquíssimos meios no Helios para contornar esse problema, especialmente quando também deseja-se obter conveniência e usabilidade para os votantes.

O protocolo do Helios, como apresentado na Figure 2, pode ser dividido em 4 fases com as respectivas primitivas criptográficas necessárias. Na fase de *configuração*, cada autoridade do protocolo submete sua chave privada e, por meio de *secret sharing*, a chave pública do sistema é construída. O esquema criptográfico utilizado é o ElGamal [ElGamal 1985], homomórfico para adição, mas existem técnicas para realização da contagem homomórfica com ele.

No *registro*, não é empregado nenhum processo criptográfico, sendo possível a eleição abrir a votação para todos os usuários do sistema, delegando o registro de cada votante para o momento da submissão da primeira cédula, ou pode ser enviada uma lista de votantes autorizados.

Durante a *votação*, o usuário é colocado em uma cabine eleitoral, devendo marcar, revisar e confirmar suas escolhas. Cada resposta da questão é uma cifra ElGamal acompanhada de uma prova de conhecimento zero que deseja provar que um valor de voto válido sem o revelar. A cédula inteira é acompanhada de uma prova de conhecimento zero.

Por fim, na *contagem*, é realizada a contagem homomórfica de todas as cédulas, as autoridades realizam a decifragem conjunta do resultado e provam por conhecimento zero que realizaram a decifragem corretamente.

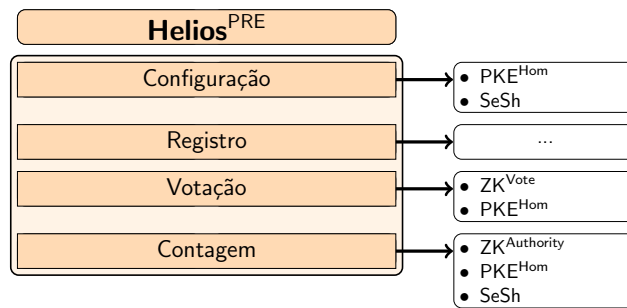


Figura 2. Fases do Helios

5. Protocolo Pós-Quântico

Esquemas criptográficos atuais são fortemente dependentes de problemas de teoria dos números, o ElGamal utilizado no Helios, por exemplo, é baseado no problema do logaritmo discreto. No entanto, esses problemas são solucionáveis pelo algoritmo de Shor [Shor 1999] quando executado em um computador quântico suficientemente poderoso. Assim, foi selecionado o protocolo de *mix-nets* pós-quântico de [Aranha et al. 2021] que além de prover segurança contra um adversário quântico, permite estruturas complexas para cédulas que vão além do n para m do Helios.

As principais primitivas do protocolo são 1) o esquema de comprometimento de [Baum et al. 2018] que é homomórfico para adição e permite instanciar provas de conhecimento zero úteis para o protocolo como a prova de *soma* (ZK^{Sum}) e especialmente a prova de *embaralhamento* (ZK^{Shuffle}) utilizada para provar que a *mix-net* decifrou os votos corretamente; e 2) o esquema de cifragem de [Lyubashevsky and Neven 2017], sendo implementada uma variação deste esquema que inclui uma prova de conhecimento que valida certas propriedades do texto cifrado, neste caso é validado que uma abertura cifrada de um comprometimento gerou este comprometimento sem a revelar. Assim, essas primitivas possuem parâmetros dependentes entre si.

Também foi implementado o mecanismo de códigos de retorno para proteção contra dispositivos maliciosos. Cada votante possui uma tabela única que utiliza uma função pseudo-aleatória PRF para mapear uma possibilidade de voto para um código. Então, o servidor no fim do processo de votação entrega um código de retorno para o votante que o compara com o resultado esperado em sua tabela.

As fases do Helios^{PQC}, apresentadas pela Figure 3, são as mesmas do Helios, no entanto, o funcionamento interno é completamente distinto, um maior nível de detalhamento das fases é apresentado na Figure 4. Na fase *configuração*, um conjunto confiável de participantes executa o algoritmo Setup para gerar as chaves do sistema dos esquemas de comprometimento e cifragem. A chave pública pk é distribuída para todos os participantes, a chave de decifração dk é dada para o Embaralhador e a chave de código ck é dada para o Gerador de Códigos de Retorno.

Na fase de *registro*, um conjunto de participantes confiáveis executam o algoritmo Register para cada votante, obtendo (vvk, vck, f) . A chave de verificação do votante vvk é pública e a chave de voto do votante vck é dada para seu computador. Em seguida, o Gerador de Códigos de Retorno seleciona uma chave k para PRF e um conjunto de partes

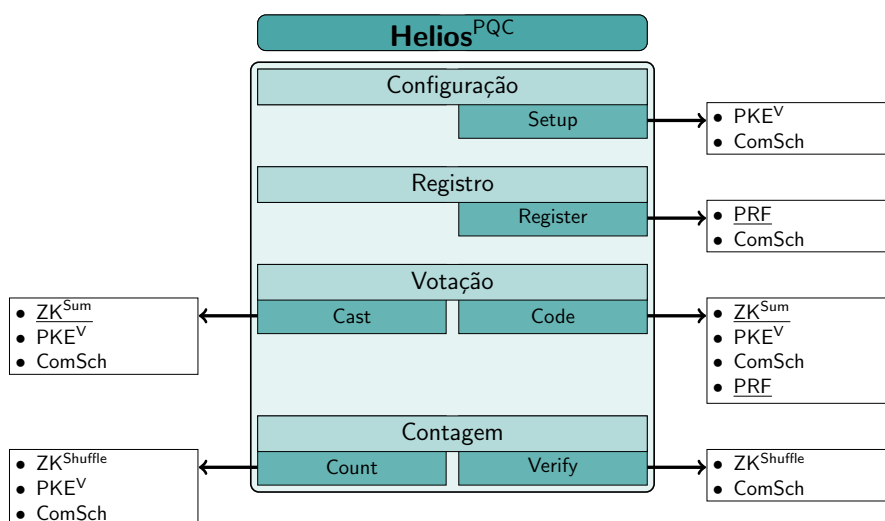


Figura 3. Fases do Helios^{PQC}

confiáveis para cada votante computam a tabela de códigos de retorno para um conjunto relativamente pequeno de cédulas e entregam-na ao votante.

Na fase de *votação*, o votante instrui seu computador qual a cédula correta de votação. Em seguida, o computador executa o algoritmo *Cast*, enviando a cédula encriptada ev e a prova da cédula Π^v para a Urna que as remete para o Gerador de Códigos de Retorno que executa o algoritmo *Code* para obter o pre-código \hat{r} , computando o código de retorno $r \leftarrow \text{PRF}_k(\hat{r})$ enviado para o celular do votante que remete para o votante que compara o código r com o código em sua tabela de códigos de retorno e apenas aceita a cédula como válida se eles são equivalentes.

Por fim, na fase de *contagem*, a Urna e o Gerador de Códigos de Retorno enviam as cédulas encriptadas e provas da cédula para os auditores. Se elas estão consistentes, os auditores aprovam. A Urna ordena as cédulas, considerando apenas os votos mais recentes de cada votante, e envia-as para o Embaralhador que usa o algoritmo *Count* para revelar e embaralhar os votos, computando a prova de embaralhamento Π^c ; ambos são enviados para os auditores, que utilizam o algoritmo *Verify* para verificar Π^c contra as cédulas encriptadas recebidas da Urna e do Gerador de Códigos de Retorno.

6. Contribuições

Neste trabalho, as primitivas criptográficas em C foram aproveitadas do artigo de [Aranha et al. 2021], no entanto, foram adicionalmente implementadas a ZK^{Sum} e a recuperação do voto a partir da abertura e comprometimento do **ComSch** e **HMAC_SHA512** foi selecionado como **PRF**. Foram mapeadas e testadas as estruturas e funções $C \rightarrow \text{Python}$ e foi implementado um protótipo para o protocolo Helios^{PQC}.

6.1. Desempenho

Para obtenção de métricas relacionadas ao desempenho e realização de uma comparação justa, as fases dos protocolos foram isoladas da interface gráfica. Foram realizados testes para os diversos procedimentos, extraindo a média de tempo apresentada em 25 vezes execuções em uma máquina de processador AMD Ryzen 7 5700X rodando em 3.4GHz.

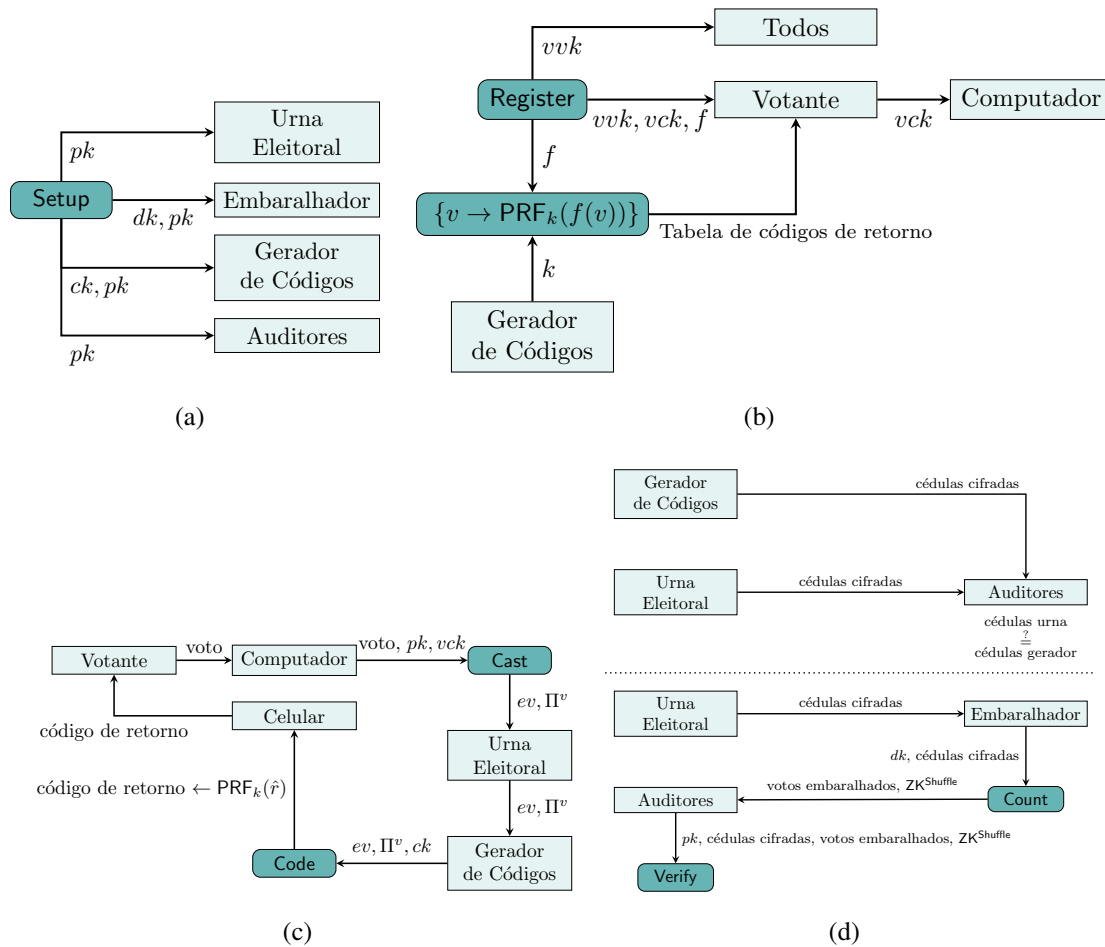


Figura 4. Fases de (a) configuração (b) registro (c) votação (d) contagem

O principal parâmetro a ser considerado é a quantidade de votantes, contudo, a quantidade de escolhas da questão também é relevante. Na Table 1, compara-se o protocolo antigo do Helios com o novo implementado. Nestes testes, foi considerada uma questão com cinco respostas, sendo pertinente destacar que a variação no número de respostas é irrelevante para o tempo de esquema de [Aranha et al. 2021], todavia, no Helios original, elas implicam diretamente no número de provas de conhecimento criadas.

Para ambos os esquemas, a fase de configuração independe do número de votantes que participaram da eleição. Observa-se que o protocolo de [Aranha et al. 2021] possui um maior desempenho que o Helios durante a fase de votação, no entanto, tem perdas significativas na fase de contagem. Neste conjunto de testes, a quantidade de votantes não foi suficientemente relevante para alterar o tempo de contagem do Helios que possui sua maior carga computacional durante a votação, já que esta fase também inclui a verificação do voto. O protocolo de [Aranha et al. 2021] por contar com o mecanismo de códigos de retorno e necessitar da prova de embaralhamento, executa operações pesadas em ambas as fases. Também é importante mencionar que como a biblioteca compartilhada é utilizada, vários processos do novo protocolo são executados em código compilado, sendo naturalmente mais veloz que a implementação em Python puro do esquema original do Helios.

	Helios			[Aranha et al. 2021]		
	25	50	100	25	50	100
Configuração	0.0054s	-	-	0.0039s	-	-
Registro	0.0002s	0.0004s	0.0008s	0.0551s	0.1374s	0.2308s
Votação	5.3263s	10.6226s	21.2814s	4.2165s	8.1156s	16.5790s
Contagem	0.0404s	0.0429s	0.0456s	0.9735s	1.8178s	3.8109s

Tabela 1. Comparação de desempenho com 25, 50 e 100 votantes entre o protocolo do Helios e o [Aranha et al. 2021]

6.2. Integração com o Sistema Helios

Para integrar o protocolo com o sistema Helios foram incluídas tabelas no banco de dados para os participantes do protocolo e serializadores foram implementados para conversão de estruturas de polinômios para JSON. A maior diferença do comportamento foi durante a contagem, de forma que, originalmente, a contagem calcula a soma entre os votos, mas mantém o resultado cifrado até a reconstrução da chave secreta das autoridade e agora o resultado já é revelado, apenas aguardando o aval do auditor para ele ser publicado.

7. Conclusão

Este trabalho atingiu o objetivo de prover uma instanciação prática de um dos protocolos de votação pós-quânticos presentes na literatura, que, no atual momento, estavam apenas apresentados em forma descritiva. Considerando o processo de contagem de votos, a mudança de um sistema homomórfico para uma *mix-net* que permite votos serem polinômios arbitrários abriu portas para a aplicação permitir eleições mais complexas. Muito se discute sobre modelos de votação que melhor representam uma população como ranqueamento e voto único transferível [ERS 2017] que agora tornam-se possíveis de serem implementados com no Helios.

Em suma, este trabalho considerou Helios como o caso de uso, contudo foi apresentada uma implementação genérica, com potencial para ser integrado em outros sistemas. Desconsiderando requisitos de infraestrutura, é relativamente simples a integração deste protocolo em outras aplicações por meio dos algoritmos desenvolvidos. Todavia, adverte-se contra o uso deste trabalho em qualquer caso no mundo real. Mesmo desconsiderando as pendências apresentadas, tanto o produto de código produzido, quanto a biblioteca utilizada como dependência são provas de conceitos com apenas fins acadêmicos. Idealmente, uma equipe especializada deve sempre ser responsável por auditar a segurança da implementação de um protocolo de votação.

7.1. Trabalhos Futuros

Existem obstáculos com um nível de complexidade inadequado para o escopo do trabalho. Assim, sugere-se como trabalhos futuros: 1) a separação dos participantes da eleição em servidores distintos; 2) a implementação de um módulo ou outra alternativa similar que permita a cifragem da cédula no lado do cliente, minimizando o aumento de complexidade da tarefa do votante; 3) o desenvolvimento de um aplicativo separado para verificação correta de códigos de retorno; e 4) a elaboração da funcionalidade de assinatura digital entre votantes com a urna e o servidor de códigos de retorno, que recentemente virou uma

possibilidade com a atualização da versão oficial do Helios por [Adida 2008] para Python 3.9.

Referências

- [Acar et al. 2018] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4).
- [Adida 2008] Adida, B. (2008). Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348.
- [Adida et al. 2023] Adida, B., de Marneffe, O., and Pereira, O. (2023). Helios election system. Online: <https://github.com/benadida/helios-server>.
- [Ali and Murray 2016] Ali, S. T. and Murray, J. (2016). An overview of end-to-end verifiable voting systems. *Real-World Electronic Voting*, pages 189–234.
- [Aranha et al. 2021] Aranha, D. F., Baum, C., Gjøsteen, K., Silde, T., and Tunge, T. (2021). Lattice-based proof of shuffle and applications to electronic voting. In *Cryptographers' Track at the RSA Conference*, pages 227–251. Springer.
- [Baum et al. 2018] Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., and Peikert, C. (2018). More efficient commitments from structured lattice assumptions. In *International Conference on Security and Cryptography for Networks*, pages 368–385. Springer.
- [Benaloh et al. 2015] Benaloh, J., Rivest, R., Ryan, P. Y., Stark, P., Teague, V., and Vora, P. (2015). End-to-end verifiability. *arXiv preprint arXiv:1504.03778*.
- [Chaum 1981] Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90.
- [Cramer et al. 1997] Cramer, R., Gennaro, R., and Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490.
- [Cranor and Cytron 1996] Cranor, L. F. and Cytron, R. K. (1996). Design and implementation of a practical security-conscious electronic polling system.
- [Dowlin et al. 2017] Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2017). Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567.
- [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.
- [ERS 2017] ERS (2017). Single Transferable Vote — [electoral-reform.org.uk](https://www.electoral-reform.org.uk). <https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/single-transferable-vote/>. [Accessed 05-06-2024].
- [Goldreich 2006] Goldreich, O. (2006). *Foundations of Cryptography: Volume 1*. Cambridge University Press, USA.

- [Goldreich and Oren 1994] Goldreich, O. and Oren, Y. (1994). Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32.
- [Goldwasser et al. 1989] Goldwasser, S., Micali, S., and Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208.
- [Halevi and Micali 1996] Halevi, S. and Micali, S. (1996). Practical and provably-secure commitment schemes from collision-free hashing. In *Annual International Cryptology Conference*, pages 201–215. Springer.
- [Kushilevitz and Ostrovsky 1997] Kushilevitz, E. and Ostrovsky, R. (1997). Replication is not needed: single database, computationally-private information retrieval. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 364–373.
- [Lyubashevsky and Neven 2017] Lyubashevsky, V. and Neven, G. (2017). One-shot verifiable encryption from lattices. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I 36*, pages 293–323. Springer.
- [Shor 1999] Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332.
- [Sun et al. 2020] Sun, X., Zhang, P., Liu, J. K., Yu, J., and Xie, W. (2020). Private machine learning classification based on fully homomorphic encryption. *IEEE Transactions on Emerging Topics in Computing*, 8(2):352–364.

9 APÊNDICE 2: CÓDIGO FONTE

- Helios modificado: <https://github.com/JoaoPedro2002/helios-server-pqc>
- Primitivas modificadas <https://github.com/JoaoPedro2002/lattice-voting-ctrsa21>