

**UNIVERSIDADE FEDERAL DE MINAS GERAIS**  
**Instituto de Ciências Exatas**  
**Programa de Pós-Graduação em Ciência da Computação**

Henrique Almeida Louzada

**Framework para Resolução de Entidades Incremental em**  
**Sistemas Federados Baseado em**  
**Criptografia Homomórfica**

Belo Horizonte  
2024

Henrique Almeida Louzada

**Framework para Resolução de Entidades Incremental em  
Sistemas Federados Baseado em  
Criptografia Homomórfica**

**Versão Final**

Dissertação apresentada ao Programa de Pós-Graduação em  
Ciência da Computação da Universidade Federal de Minas  
Gerais, como requisito parcial à obtenção do título de Mestre  
em Ciência da Computação.

Orientador: Wagner Meira Junior

Belo Horizonte  
2024

Louzada, Henrique Almeida.

L895f      Framework para resolução de entidades incremental em sistemas federados baseado em criptografia homomórfica [recurso eletrônico] / Henrique Almeida Louzada – 2024.  
1 recurso online (70 f. il., color.) : pdf.

Orientador: Wagner Meira Júnior.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.

Referências: f. 61-64.

1. Computação – Teses. 2. Arquitetura de computador – Teses. 3. Arquitetura de Gestão Federada (Arquitetura de Computadores) – Teses. 4. Criptografia de dados (Computação) – Teses. 5. Resolução de Entidades - Teses. I. Meira Júnior, Wagner. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6\*21(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

## FOLHA DE APROVAÇÃO

Framework para Resolução de Entidades Incremental em Sistemas  
Federados Baseado em Criptografia Homomórfica

**HENRIQUE ALMEIDA LOUZADA**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

A handwritten signature in blue ink, appearing to read "Wagner Meira Junior".

Prof. Wagner Meira Júnior  
Departamento de Ciência da Computação - UFMG

A handwritten signature in blue ink, appearing to read "Aldri Luiz dos Santos".

Prof. Aldri Luiz dos Santos  
Departamento de Ciência da Computação - UFMG

A handwritten signature in blue ink, appearing to read "Altigran Soares da Silva".

Prof. Altigran Soares da Silva  
Instituto de Computação - UFAM

Belo Horizonte, 3 de dezembro de 2024.

*Dedicado a Iara Soares Silva (in memoriam). Avó, mãe,  
educadora.*

# Agradecimentos

Mesmo antes de uma fatalidade tão inoportuna acontecer na reta final do meu sonho de mostrar a ela que todo o seu sacrifício valeu a pena, meu desejo sempre foi dedicar esta dissertação à pessoa que me formou como aluno, profissional e ser humano. Iara foi uma avó extraordinária, que dedicou sua vida ao próximo — seja a seus incontáveis alunos ao longo de sua carreira como professora, seja à sua família, à qual serviu até seus últimos dias neste mundo.

Este projeto é o testemunho máximo da minha resiliência, motivada pela trajetória irreparável e vitoriosa de uma mulher que suportou a dor devastadora de perder um filho, de ser abandonada com três crianças, de envelhecer com a dor da depressão, e de criar gerações de descendentes sozinha. Sua capacidade de encontrar propósito em meio a tantas adversidades foi o que me inspirou a ter forças para não desistir quando sua presença física nos deixou no início do meu curso de mestrado, restando apenas seus valiosos ensinamentos e o seu espírito, que ainda vive radiante em meu coração. Muito obrigado, Dona Iara.

Este trabalho foi parcialmente financiado pelas agências CNPq, CAPES e FAPEMIG, e pelos projetos CIIA-Saúde e IAIA-INCT em IA.

*“Under the bludgeonings of chance  
My head is bloody, but unbowed.”*  
(W. E. Henley)

# Resumo

A Resolução de Entidades é o tópico da computação que estuda maneiras otimizadas de identificar diferentes registros em fontes de dados e classificá-los como representantes de um mesmo elemento. A realização dessa operação em contextos reais implica em uma série de subtemas, incluindo questões que dizem respeito ao cenário de aplicação, como a privacidade das entidades envolvidas.

A arquitetura de Sistemas Federados propõe o deslocamento do ambiente de processamento de um procedimento computacional específico para um ou mais servidores apartados, permitindo que vários clientes colaborem em um objetivo comum. Esse tipo de arquitetura é útil à Resolução de Entidades pois algumas de suas aplicações têm como objetivo o cruzamento de dados de diversas fontes, o que acaba ocasionando a necessidade de um compartilhamento protegido desses dados.

Este trabalho investiga o uso da Criptografia Homomórfica para Resolução de Entidades usando Computação Federada, propondo um *framework* que permite a verificação de correspondências entre registros criptografados sem que seja necessário descriptá-los. O *framework* é implementado em linguagem de alto nível, e suporta inclusive técnicas de blocagem sob criptografia, reduzindo a complexidade computacional sem comprometer a qualidade da análise.

Ao experimentar o *framework* com 11.327 tuplas de dados demográficos de pacientes oncológicos criados artificialmente, é possível observar que diferentes configurações de blocagem afetam diretamente as métricas de desempenho e qualidade das resoluções, chegando a apresentar uma Taxa de Redução de 91% em certos cenários e uma Completude de Pares majoritariamente acima dos 80%. É possível notar também que a Resolução de Entidades, mesmo limitada pelas opções de operações homomórficas, chega a 63,03% de Coeficiente de Jaccard e 77,32% de F1-Score. Em termos de desempenho, o tempo de execução da Resolução de Entidades não é afetado pela Criptografia Homomórfica, mas apresenta melhoria significativa com a estratégia de blocagem. O estudo também explora possíveis melhorias para o *framework*.

**Palavras-chave:** resolução de entidades; ligação de registros; sistemas federados; criptografia homomórfica.

# Abstract

Entity Resolution is the computing topic that studies optimized ways to identify different records in data sources and classify them as representatives of the same element. Performing this operation in real contexts involves a series of subtopics, including issues related to the application scenario, such as the privacy of the entities involved.

The Federated Systems architecture proposes moving the processing environment of a specific computational procedure to one or more separate servers, allowing multiple clients to collaborate towards a common goal. This type of architecture is useful for Entity Resolution because some of its applications aim to cross-reference data from multiple sources, which ends up causing the need for secure sharing of this data.

This work investigates the use of Homomorphic Encryption for Entity Resolution using Federated Computing, proposing a framework that allows the verification of correspondences between encrypted records without the need to decrypt them. The framework is implemented in a high-level language and even supports encryption blocking techniques, reducing computational complexity without compromising the quality of the analysis.

When testing the framework with 11,327 artificially created tuples of demographic data of oncology patients, it is possible to observe that different blocking configurations directly affect the performance and quality metrics of the resolutions, reaching a Reduction Rate of 91% in certain scenarios and a Pairwise Completeness mostly above 80%. It is also possible to note that the Entity Resolution, even limited by the options of homomorphic operations, reaches 63.03% in the Jaccard Coefficient and 77.32% in F1-Score. In terms of performance, the runtime of Entity Resolution is not affected by the Homomorphic Encryption, but shows significant improvement with the blocking strategy. The study also explores possible improvements for the *framework*.

**Keywords:** entity resolution; record linkage; federated systems; homomorphic encryption.

# Lista de Figuras

|     |                                                                                                                                                                                                                                                                  |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1 | Exemplificação da estratégia de Blocação Representativa. A seleção de representantes é importante para o cenário criptografado do <i>framework</i> pois reduz a quantidade de operações criptográficas necessárias para posicionar um registro entrante. . . . . | 29 |
| 3.2 | Ilustração da redução de complexidade promovida pela Blocação Representativa na tarefa da Resolução de Entidades Incremental. . . . .                                                                                                                            | 30 |
| 3.3 | Transações entre cliente e servidor do <i>framework</i> . . . . .                                                                                                                                                                                                | 38 |
| 4.1 | Variação da métrica de Completude de Pares $CP(\mathcal{B})$ entre diferentes números de atributos do critério de resolução. . . . .                                                                                                                             | 52 |
| 4.2 | Variação da métrica de Qualidade de Pares $QP(\mathcal{B})$ entre diferentes números de atributos do critério de resolução. . . . .                                                                                                                              | 52 |
| 4.3 | Variação da métrica de Taxa de Redução $TR(\mathcal{B}, \mathcal{E})$ entre diferentes números de atributos do critério de resolução. . . . .                                                                                                                    | 53 |
| 4.4 | Matriz de Confusão da RE com 7 de 11 campos como critério de resolução. O quadrante mais escuro reforça a presença majoritária de registros sem réplica e que assim foram classificados pelo algoritmo de RE. . . . .                                            | 55 |
| 4.5 | Tempo de execução em segundos para uma operação de RE em função do número de valores em comum, para os quatro cenários descritos. . . . .                                                                                                                        | 58 |

# Lista de Tabelas

|      |                                                                                                                                                                                                                                                                                 |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1  | Apresentação do conjunto de dados. . . . .                                                                                                                                                                                                                                      | 27 |
| 3.2  | Subtração Homomórfica entre <i>strings</i> de tamanhos iguais. . . . .                                                                                                                                                                                                          | 33 |
| 3.3  | Conjuntos de dados alocados pelo <i>framework</i> em tempo de execução. . . . .                                                                                                                                                                                                 | 37 |
| 3.4  | Amostra do conjunto de dados <i>ti</i> . . . . .                                                                                                                                                                                                                                | 41 |
| 3.5  | Conjuntos de dados <i>ti</i> e <i>rr</i> , resultantes do processo de eleição de representantes do Algoritmo 3.3. . . . .                                                                                                                                                       | 41 |
| 3.6  | Conjuntos de dados <i>eti</i> e <i>err</i> , resultantes do processo de encriptação do Algoritmo 3.4. Os sinais de '[' e ',' denotam uma coleção de cifras geradas aleatoriamente, semelhantes à primeira descrita pelo valor da célula. . . . .                                | 42 |
| 3.7  | <i>ete</i> é criado a partir da encriptação da tupla entrante <i>te</i> . . . . .                                                                                                                                                                                               | 43 |
| 3.8  | Amostra do conjunto de dados cifrados <i>esrr</i> , criado a partir da subtração homomórfica do Algoritmo 3.7. . . . .                                                                                                                                                          | 43 |
| 3.9  | Amostra do conjunto de dados <i>srr</i> , resultante da escoragem do algoritmo 3.8. As colunas 'pontos' e 'bloco' mostram que o bloco 1 contabilizou 8 colunas de valores iguais à tupla entrante <i>te</i> , sendo o bloco selecionado pela sua pontuação mais alta. . . . .   | 44 |
| 3.10 | Amostra do conjunto de dados <b>cifrados</b> <i>esti</i> , resultante da subtração homomórfica do Algoritmo 3.9. . . . .                                                                                                                                                        | 44 |
| 3.11 | Amostra da escoragem realizada pelo Algoritmo 3.10. A coluna 'pontos' mostra que a tupla de índice 31 contabilizou 10 colunas de valores iguais à tupla entrante <i>te</i> , sendo o registro que satisfaz o requisito da função-oráculo para a resolução de entidades. . . . . | 45 |
| 4.1  | Apresentação do conjunto de dados. . . . .                                                                                                                                                                                                                                      | 46 |
| 4.2  | Dados oncológicos sintetizados utilizados no trabalho. . . . .                                                                                                                                                                                                                  | 47 |
| 4.3  | Representante do grupo "ARAUJO", no cenário de blocagem por "nomeu". O valor das colunas representa o valor mais recorrente da coluna no conjunto de dados original. . . . .                                                                                                    | 48 |
| 4.4  | Métricas de qualidade da Blocagem Representativa para agrupamentos de 1 coluna. . . . .                                                                                                                                                                                         | 49 |
| 4.5  | Métricas de qualidade da Blocagem Representativa para agrupamentos de 2 colunas. . . . .                                                                                                                                                                                        | 50 |

|     |                                                                                                                                                                                                                                                                                                               |    |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.6 | Métricas de qualidade da Blocagem Representativa para diferentes valores do critério de resolução. Cada coluna de valores numéricos representa a quantidade mínima de atributos idênticos para considerar um par de registros como representantes de uma mesma entidade, variando de 5 a 9 atributos. . . . . | 51 |
| 4.7 | Detalhamento das métricas componentes de $QP(\mathcal{B})$ . Tanto $ \mathcal{D}(\mathcal{B}) $ quanto $  \mathcal{B}  $ decaem com o aumento do limiar de campos, porém a proporção entre essas duas métricas não se mantém constante. . . . .                                                               | 54 |

# Lista de Algoritmos

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1  | Pseudocódigo com a implementação da classe Cliente. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 39 |
| 3.2  | Pseudocódigo com a implementação da classe Servidor. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 40 |
| 3.3  | Instanciação do cliente Bob e chamada ao método de eleição de representantes. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                | 41 |
| 3.4  | Os conjuntos $ti$ e $rr$ passam pelo processo de encriptação utilizando o sistema de Paillier. . . . .                                                                                                                                                                                                                                                                                                                                                                                                               | 41 |
| 3.5  | Instanciação do servidor Alice. $sk$ não é compartilhada com o servidor, o que garante que qualquer dado compartilhado com Alice seja incapaz de ser descriptado. . . . .                                                                                                                                                                                                                                                                                                                                            | 42 |
| 3.6  | Instanciação do cliente Carol e encriptação da tupla entrante $te$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                         | 43 |
| 3.7  | Subtração homomórfica do conjunto de dados realizada entre $err$ e $ete$ pelo servidor Alice, resultando em $esrr$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                         | 43 |
| 3.8  | Escoragem realizada pelo cliente <i>Carol</i> a partir de $esrr$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                           | 44 |
| 3.9  | Chamada à subtração homomórfica realizada entre $eti$ e $ete$ , resultando em $esti$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                       | 44 |
| 3.10 | Escoragem realizada pelo cliente <i>Carol</i> a partir de $esti$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                           | 45 |
| A.1  | Pseudocódigo com a implementação da classe Paillier. A função <i>gerar_chaves</i> cria duas chaves, uma pública $pk$ composta por $n$ e $g$ , e outra privada composta por $\lambda$ e $\mu$ , conforme definição do algoritmo. . . . .                                                                                                                                                                                                                                                                              | 65 |
| A.2  | Continuação do pseudocódigo com a implementação da classe Paillier. A função <i>encriptar</i> consulta os valores contidos na chave pública, gera um número aleatório $r$ e calcula a cifra do texto original informado por parâmetro através da fórmula dada por definição. A função <i>desencriptar</i> faz a operação reversa, consultando os valores das chaves pública e privada, e recuperando o texto original a partir da cifra informada por parâmetro ao implementar a fórmula dada por definição. . . . . | 66 |
| A.3  | Continuação do pseudocódigo com a implementação da classe Paillier. A função <i>subtracao_homomorfica</i> realiza a multiplicação modular entre uma primeira cifra informada por parâmetro e o inverso modular de uma segunda cifra, também informada por parâmetro. . . . .                                                                                                                                                                                                                                         | 67 |

# Sumário

|          |                                                                                                        |           |
|----------|--------------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introdução</b>                                                                                      | <b>15</b> |
| 1.1      | Motivação . . . . .                                                                                    | 16        |
| 1.2      | Objetivo Geral e Específicos . . . . .                                                                 | 16        |
| 1.2.1    | Objetivo Geral . . . . .                                                                               | 16        |
| 1.2.2    | Objetivos Específicos . . . . .                                                                        | 16        |
| 1.3      | Contribuições pretendidas . . . . .                                                                    | 17        |
| 1.4      | Organização . . . . .                                                                                  | 17        |
| <b>2</b> | <b>Referencial Teórico</b>                                                                             | <b>18</b> |
| 2.1      | Resolução de Entidades e Problemas Relacionados . . . . .                                              | 18        |
| 2.1.1    | Blocagem e Filtragem . . . . .                                                                         | 19        |
| 2.1.2    | Resolução de Entidades Incremental . . . . .                                                           | 19        |
| 2.1.3    | Aplicações e Contribuições . . . . .                                                                   | 20        |
| 2.1.4    | O Problema da Privacidade . . . . .                                                                    | 21        |
| 2.2      | Computação Federada . . . . .                                                                          | 22        |
| 2.3      | Criptografia Homomórfica . . . . .                                                                     | 23        |
| 2.4      | Trabalhos Relacionados . . . . .                                                                       | 24        |
| 2.5      | Sumário . . . . .                                                                                      | 26        |
| <b>3</b> | <b>Framework para Resolução de Entidades Homomorficamente Encrip-<br/>tadas em Ambientes Federados</b> | <b>27</b> |
| 3.1      | Estratégia de Blocagem . . . . .                                                                       | 28        |
| 3.1.1    | Métricas Fundamentais . . . . .                                                                        | 28        |
| 3.1.2    | Análise da Qualidade de Blocagem . . . . .                                                             | 29        |
| 3.2      | Resolução de Entidades Sob Criptografia Homomórfica . . . . .                                          | 31        |
| 3.2.1    | Sistema Criptográfico de Paillier . . . . .                                                            | 32        |
| 3.2.2    | Análise da Qualidade de Resolução de Entidades . . . . .                                               | 34        |
| 3.3      | Ambiente de Computação Federado . . . . .                                                              | 35        |
| 3.3.1    | Servidor de Processamento Centralizado . . . . .                                                       | 35        |
| 3.3.2    | Clientes Colaboradores . . . . .                                                                       | 36        |
| 3.4      | Implementação . . . . .                                                                                | 37        |
| 3.4.1    | Código-fonte . . . . .                                                                                 | 38        |
| 3.4.2    | Exemplo de Implementação . . . . .                                                                     | 38        |

|          |                                                                               |           |
|----------|-------------------------------------------------------------------------------|-----------|
| 3.4.2.1  | Instância do cliente Bob e alocação de dados na memória .                     | 40        |
| 3.4.2.2  | Encriptação de $ti$ e $rr$ . . . . .                                          | 41        |
| 3.4.2.3  | Criação do ponto inicial de contribuição . . . . .                            | 42        |
| 3.4.2.4  | Preparação da Tupla Entrante . . . . .                                        | 42        |
| 3.4.2.5  | Subtração homomórfica e pontuação da Relação de Representantes $rr$ . . . . . | 43        |
| 3.4.2.6  | Subtração homomórfica e pontuação da Tabela Integral $ti$ .                   | 44        |
| <b>4</b> | <b>Resultados Experimentais</b>                                               | <b>46</b> |
| 4.1      | Caracterização dos Dados . . . . .                                            | 46        |
| 4.2      | Blocagem Representativa . . . . .                                             | 47        |
| 4.2.1    | Cenários de Blocagem por Conjuntos de Colunas . . . . .                       | 47        |
| 4.2.2    | Variação do Critério de Resolução . . . . .                                   | 50        |
| 4.3      | Resolução de Entidades . . . . .                                              | 54        |
| 4.3.1    | Coefficiente de Jaccard . . . . .                                             | 55        |
| 4.3.2    | Precisão . . . . .                                                            | 56        |
| 4.3.3    | Revocação . . . . .                                                           | 56        |
| 4.3.4    | F1-Score . . . . .                                                            | 57        |
| 4.3.5    | Desempenho Computacional . . . . .                                            | 57        |
| <b>5</b> | <b>Conclusão e Trabalhos Futuros</b>                                          | <b>59</b> |
|          | <b>Referências</b>                                                            | <b>61</b> |
|          | <b>Apêndice A Sistema Criptográfico de Paillier - Implementação</b>           | <b>65</b> |
|          | <b>Apêndice B Sistema Criptográfico de Paillier - Corretude</b>               | <b>68</b> |
| B.1      | Segurança da Criptografia e Garantia de Homomorfismo . . . . .                | 68        |
| B.2      | Corretude da Descriptografia . . . . .                                        | 69        |

# Capítulo 1

## Introdução

A descoberta de conhecimento em bancos de dados é um campo da computação que se concentra na extração de informações úteis a partir de grandes conjuntos de dados. Esse processo envolve a aplicação de técnicas e algoritmos de análise, como mineração de dados e inteligência artificial, para identificar relações complexas entre os dados, possibilitando a geração de *insights* valiosos para a tomada de decisão e a previsão de eventos futuros. Com isso, é possível, por exemplo, revelar padrões de comportamento de clientes de um produto específico, identificar fraudes e otimizar processos industriais.

A etapa de tratamento de dados é parte fundamental do processo de descoberta de conhecimento em bancos de dados, especialmente com a crescente quantidade de insumos à disposição do usuário final. É neste momento, anterior ao processamento do conjunto em estudo, que procedimentos como amostragem, transformação e limpeza são utilizados com a finalidade de reduzir o gasto de recursos computacionais sem impacto negativo significativo no resultado final da análise. Em um nível mais detalhado, a Resolução de Entidades (RE) é um procedimento de tratamento de dados que consiste na identificação de registros não idênticos que representam uma mesma entidade no mundo real, além de uni-los em um único registro segundo critérios arbitrários. Isso pode ser feito por meio de uma comparação direta entre todos os possíveis pares de tuplas ou por meio de técnicas de agrupamento. Com a RE, é possível não apenas simplificar o objeto de estudo, mas também realizar outras tarefas relacionadas, como garantir a rastreabilidade de tuplas e a desambiguação de informações no próprio domínio real do problema [5].

É importante ressaltar, contudo, que introduzir a RE na realidade do mercado levanta questões que desafiam a viabilidade dessa técnica. Escolher entre uma varredura completa ou utilizar técnicas de agrupamento apresenta prós e contras: intuitivamente, a primeira é mais custosa, porém mais confiável; já a segunda é menos custosa, mas traz consigo uma margem de erro. Esse problema é abordado pela literatura por meio da sofisticação e otimização das técnicas de agrupamento, buscando reduzir tal margem de erro sem comprometer a confiabilidade da solução [10].

## 1.1 Motivação

Uma questão particular nos domínios de aplicação da RE é a privacidade dos dados analisados. Muitas vezes, o contexto em questão envolve conjuntos de dados sensíveis, como, por exemplo, registros médicos de pacientes, informações de clientes de empresas privadas ou registros de atividades de usuários. A arquitetura dos sistemas desse tipo de negócio pode, inclusive, interligar duas fontes de dados distintas através da rede, o que torna o processo suscetível a interceptações. O vazamento desse tipo de informação pode causar prejuízos, e por isso a preocupação com a sua privacidade está se tornando latente em trabalhos na área da RE [14].

Este problema é abordado pelo estado da arte através de diversas técnicas, como por exemplo os *frameworks* que utilizam ofuscação com a adição de dados falsos e manipulação do domínio [13], ou mesmo *frameworks* de Resolução de Entidade às cegas (ou *Blindfolded Entity Resolution*) [12]. Porém, quando arquiteturas computacionais mais modernas são introduzidas para a implementação desses sistemas, é necessário que o sistema criptográfico também seja melhor elaborado para ser capaz de continuar garantindo performance e segurança no procedimento de RE [13].

## 1.2 Objetivo Geral e Específicos

### 1.2.1 Objetivo Geral

O objetivo geral deste trabalho é propor, projetar e avaliar um *framework* para resolução de entidades em sistemas federados, garantindo a privacidade, a efetividade e o desempenho utilizando criptografia homomórfica.

### 1.2.2 Objetivos Específicos

1. Propor e discutir o uso da criptografia homomórfica no procedimento de Resolução de Entidades, a partir da premissa que propiciaria uma redução da complexidade

do pré-processamento desses dados, ao mesmo tempo que provê uma camada de segurança para dados trafegados em rede;

2. Projetar e implementar um *framework* de Computação Federada, o que, com o auxílio de operações de criptografia homomórfica, torna possível o processamento da Resolução de Entidades mesmo quando os dados se encontram criptografados [15].
3. Validar e avaliar experimentalmente o *framework* proposto com bases significativas.

## 1.3 Contribuições pretendidas

O trabalho contribui com a formulação do *framework* de Resolução de Entidades incremental com criptografia homomórfica, a sua implementação em código-fonte (publicado na plataforma GitHub e disponível em [17]), e a análise de seu desempenho em termos de qualidade de agrupamento e performance computacional.

## 1.4 Organização

O Capítulo 2 apresenta um referencial com a contextualização de cada componente teórico do trabalho, o Capítulo 3 discorre sobre o desenvolvimento do *framework* e o Capítulo 4 mostra os resultados observados a partir das métricas obtidas. Por fim, o Capítulo 5 conclui o trabalho e pontua próximos passos e possíveis projetos relacionados a esta iniciativa.

## Capítulo 2

# Referencial Teórico

A Resolução de Entidades tem como tarefa principal a desambiguação de registros, e se difere da Vinculação de Registros (ou *Record Linkage*) pelo seu desfecho, que acontece com a criação de um vínculo lógico entre as entidades relacionadas, mantendo ambas em seu repositório original. Isso não acontece na RE, que acaba substituindo os registros por uma mescla que melhor representa a entidade real a qual eles se referem. Diversos outros termos são associados a este tipo de procedimento, como a Deduplicação, ou Reconciliação de Referência [32]. Como este trabalho se concentra na tarefa inicial de desambiguação, comum a estes procedimentos, a sua literatura também é consultada.

## 2.1 Resolução de Entidades e Problemas Relacionados

O processo da RE pode variar de acordo com sua implementação. A solução de força bruta em um cenário com dados estruturados se trata de uma comparação direta e exata entre dois valores de mesma dimensão em diferentes tuplas. A partir disso, as evoluções são abundantes e constantes. Alguns exemplos são a medida de distância de Levenshtein entre os valores, o uso de fonemas representativos e o agrupamento de registros. O uso de fontes secundárias também pode ser útil para o agrupamento de registros em domínios específicos [5]. Outras abordagens focam na redução da complexidade computacional envolvida na RE. Nesse sentido, as estratégias de blocagem e filtragem podem ser consideradas como uma etapa de pré-seleção dos candidatos antes da RE propriamente dita.

### 2.1.1 Blocagem e Filtragem

Visando a redução de complexidade do problema de RE, suas implementações geralmente consistem em duas partes, sendo uma pré-seleção de candidatos e a resolução propriamente dita, já considerando somente aqueles que valem a pena serem comparados. Tal pré-seleção é majoritariamente realizada através das estratégias de blocagem e filtragem. As evoluções na área de RE muitas vezes experimentam maneiras diferentes de aumentar a sua *efetividade* e a sua *eficiência*, definidas respectivamente pela capacidade de resolver entidades corretamente e qual o nível de esforço computacional necessário para realizar essa tarefa, o que é convenientemente o objetivo dos processos pré-seletivos [22].

A complexidade da RE é naturalmente considerada quadrática se levarmos em consideração a comparação de cada par possível de registros em um conjunto de dados qualquer. O procedimento de blocagem tem como objetivo reduzir essa complexidade através do agrupamento de registros por características arbitrariamente semelhantes, fazendo com que registros de grupos diferentes nem sequer sejam incluídos no processo de comparação. A filtragem, por sua vez, acontece em sequência à blocagem, e utiliza um limiar de semelhança para excluir verdadeiros negativos do conjunto candidato. Existem diversas estratégias para a construção dos blocos, e podem variar entre si por dimensões, como por exemplo a de tipo de chaveamento, que dita se a estratégia de blocagem considera a equidade ou a semelhança de uma chave comum a dois registros distintos [22]. A necessidade de lidar com a alta complexidade da Resolução de Entidades (RE), especialmente em grandes conjuntos de dados, tem levado ao desenvolvimento de estratégias como a blocagem e a filtragem para reduzir o número de comparações necessárias. No entanto, esses métodos tradicionais enfrentam novos desafios à medida que os conjuntos de dados se tornam dinâmicos e em constante mudança.

Para lidar com essa realidade, surge a Resolução de Entidades Incremental, uma abordagem que se fundamenta nos conceitos de blocagem e filtragem, mas que é adaptada para otimizar o processo de agrupamento em cenários onde os dados são continuamente atualizados. A subseção a seguir discorre sobre a técnica.

### 2.1.2 Resolução de Entidades Incremental

A abordagem incremental da Resolução de Entidades é necessária para evitar a reexecução completa do processo de agrupamento, o que seria ineficiente com grandes volumes de dados. Em vez disso, a RE incremental visa atualizar apenas as partes relevantes

dos resultados anteriores para refletir as mudanças nos dados, mantendo a precisão e a eficiência do processo. Isso envolve a identificação de registros afetados pelas mudanças nos dados e a atualização dos grupos de entidades conforme necessário [10, 32].

A RE em seu estágio incremental foi abordada por este trabalho para garantir suporte tanto à operação inicial de formação dos dados compartilhados quanto às consultas feitas pelos contribuintes da rede. Neste caso, a redução da complexidade de comparações integrais para incrementais seria de  $O(n^2)$  para  $O(n)$ . Desconsiderar o cenário integral em detrimento do incremental significa deixar de atestar repetidas vezes sua efetividade, razão pela qual ele foi abreviado.

Para atender ao problema da Resolução de Entidades Incremental, *frameworks* mais recentes propõem soluções em grafos. [27] formularam um algoritmo capaz de realizar a resolução incremental 33% mais rápido do que o estado da arte, modelando as relações entre registros similares através de um grafo, que é analisado repetidamente para encontrar componentes conectados. Cada um desses componentes representa um registro referente a uma entidade única do mundo real. [6] pontuam vantagens e desvantagens de diversas estratégias de Resolução de Entidades no contexto do *Big Data*, partindo de um grafo de dependências para realizar a blocagem dos registros.

Este trabalho busca emular uma situação de RE incremental, onde o usuário insere um novo registro na base de dados original, buscando ali tuplas que possam ser representantes da mesma entidade que o registro entrante. Dessa forma, nos concentramos na técnica de RE de maneira mais atômica, reduzindo a complexidade de processamento quando se comparado à RE de uma base integral, onde todas as tuplas são testadas umas contra as outras em busca de réplicas. A seção a seguir exemplifica diversos cenários de utilidade para a RE.

### 2.1.3 Aplicações e Contribuições

Tomando um panorama geral da presença da RE na literatura, [3] utilizaram modelos estatísticos e algoritmos determinísticos para realizar a desambiguação de famílias nos registros de taxação anual do Cabo da Boa Esperança, e acabaram obtendo ligações de boa qualidade para 84% do conjunto de dados ao validar com dados interligados manualmente. [29] utilizaram três algoritmos de aprendizado de máquina combinados para melhorar o desempenho da RE referentes a pacientes situados em diferentes serviços, e chegaram a melhorar a precisão da classificação em mais de 37% em uma das bases de dados estudadas. [20] compararam o uso de algoritmos probabilísticos e determinísticos na tarefa de vinculação de registros de mortalidade com histórico de prescrição medi-

camentosa do programa de monitoramento do Tennessee, e observaram um desempenho semelhante entre as duas estratégias. [18] propuseram um classificador de municípios e bancos através de uma rede de perceptrons multicamadas, retirando a dependência de implementações baseadas em limiares.

Alinhada às necessidades do mundo real, a RE é uma técnica de análise de dados que pode auxiliar na conclusão de tarefas de diversas áreas do conhecimento. Este trabalho utiliza um conjunto de dados sintético, inspirado em dados de pacientes oncológicos. Este é um problema real que acomete pacientes que, por exemplo, possuem seu prontuário vinculado a cooperativas de planos de saúde privados e, após diagnóstico positivo para câncer, criam novo registro na base de saúde governamental para darem sequência ao tratamento. A falta de padronização dos formulários de cada prontuário dificulta o acompanhamento da evolução do quadro clínico com o passar do tempo. A RE é introduzida nesse contexto com o intuito de unificar esses registros de diferentes bases, criando assim um prontuário mais detalhado deste paciente.

Existe, porém, uma preocupação com a privacidade dos dados neste processo. Muitas vezes a RE é realizada em servidores apartados das entidades que custodiam os dados utilizados. Nesse sentido, o problema da exposição de dados sensíveis na RE – de prontuários médicos, por exemplo – também é abordado pela literatura.

#### 2.1.4 O Problema da Privacidade

O problema da privacidade no âmbito da RE surge de casos de uso reais como uma preocupação com os impactos que a exposição de um registro causaria para quem o mantém e para a entidade que ele representa. [13] apresenta 4 projetos de privacidade para a RE, entre eles a dissociação de dados, que significa encriptar as informações e criar protocolos aceitáveis de revelação no momento em que elas forem necessárias. Mais recentemente, para a Resolução de Entidades Incremental, [12] empregaram uma estratégia de anonimização de dados através da codificação fonética (para campos textuais) e a K-anonimização (para campos numéricos e categóricos), e provaram que é possível realizar a RE com os dados nesse estado, em troca de uma leve perda de qualidade.

Tendo ambas essas perspectivas em vista, empregar a estratégia de criptografia homomórfica utilizada por [15] significa unir os benefícios da anonimização proposta por [12] com a utilidade dissociativa da criptografia: "Os dados desacoplados fornecem o mesmo nível de proteção que os dados desidentificados, mas com mais proteção do que é fornecido no modo de acesso confiável e mais utilidade"[13].

## 2.2 Computação Federada

A Computação Federada se trata de um tipo de arquitetura de computação através da qual diversos colaboradores conseguem somar esforços para atingir um objetivo em comum. Nessa arquitetura, cada participante detém o controle de seus dados e as tarefas computacionais são distribuídas entre uma ou mais unidades de processamento da rede. Essa abordagem permite o compartilhamento de recursos sem a necessidade de centralizar dados, aumentando assim a privacidade, a segurança e a escalabilidade de ambientes computacionais distribuídos. Algumas tarefas comumente realizadas utilizando Computação Federada incluem a análise de dados privada e o Aprendizado Federado, que se trata de uma abordagem que permite treinar modelos de aprendizado de máquina de forma descentralizada, mantendo os dados na máquina dos usuários ao invés de centralizá-los em um único servidor [8].

Por se tratar de uma arquitetura que envolve o armazenamento e processamento de dados e também da comunicação entre unidades computacionais, logo surge a necessidade de se discutir a respeito da cibersegurança em soluções de Computação Federada. Alguns exemplos de questões que circundam a cibersegurança dessa arquitetura incluem a acessibilidade às complexas implementações de computação segura, a complexidade de integração em sistemas legados e a otimização da solução dada a restrição de recursos disponíveis. Trabalhos mais recentes buscam a facilitação da implementação desses protocolos de segurança para que assim se tornem abrangentes com usuários mais precarizados como *startups* e institutos de pesquisa [4], além da sua otimização em aplicações mais específicas, como no *Edge Computing*, *Blockchain* e no *Machine Learning* [33, 15, 31].

Trabalhar com arquiteturas federadas no contexto médico significa habilitar que, por exemplo, diversos centros médicos contribuam com registros de seus pacientes para um servidor comum, onde ocorrem operações de transformação e geração de valor a partir desses dados [8]. Isso torna possível que tal valor possa ser aproveitado por cada um dos participantes dessa rede sem que de fato eles precisem possuir um volume tão expressivo de dados. Tendo isso em vista, este trabalho busca replicar um cenário de RE de pacientes oncológicos em um ambiente computacional federado, o que eleva a complexidade do problema da privacidade citado anteriormente na Seção 2.1.4. Isso acontece pois agora, além de demandar que os dados precisem ser processados em um servidor apartado, também é necessário que essa operação seja feita inteiramente de maneira sigilosa. A Criptografia Homomórfica surge nesse contexto como uma solução para o processamento de dados sob criptografia, sem a necessidade de decodificação [21].

## 2.3 Criptografia Homomórfica

A Criptografia Homomórfica é uma forma especializada de criptografia que permite que certos tipos de cálculos sejam realizados em dados criptografados sem a necessidade de descriptografá-los. Esta propriedade permite que os cálculos realizados em texto cifrado produzam um resultado criptografado que corresponde ao resultado das operações realizadas no texto original [21].

Existem diferentes níveis de esquemas de criptografia homomórfica:

- Criptografia Parcialmente Homomórfica (ou *Partially Homomorphic Encryption*), com a qual é possível realizar operações de adição ou multiplicação em dados criptografados, mas não ambas.
- Criptografia um Tanto Homomórfica (ou *Somewhat Homomorphic Encryption*), que pode suportar múltiplas operações, incluindo adição e um número limitado de operações de multiplicação.
- Criptografia Totalmente Homomórfica (ou *Fully Homomorphic Encryption*), que pode realizar operações de adição e multiplicação, permitindo o cálculo de qualquer função em dados criptografados.

A principal vantagem da criptografia homomórfica é que ela permite que cálculos sejam realizados em dados confidenciais enquanto eles permanecem criptografados, preservando assim a privacidade e a segurança das informações. Esta capacidade é particularmente valiosa em cenários onde os dados precisam ser processados ou analisados por terceiros sem comprometer a sua confidencialidade [21].

A criptografia homomórfica é capaz de trazer impactos positivos para vários campos, incluindo a computação em nuvem, análise de dados privada, agregação privada de aprendizado de máquina e processamento seguro de dados de IoT. Técnicas de criptografia totalmente homomórfica já são inclusive utilizadas em cenários de *big data* para garantir a privacidade de operações que envolvem aprendizado de máquina [23]. É importante destacar, porém, que suas implementações mais atuais enfrentam vários desafios que impactam sua praticidade e, consequentemente, sua popularização. Alguns dos principais desafios incluem a sobrecarga computacional causada pelas operações matemáticas complexas da Criptografia Homomórfica, o gerenciamento do crescimento de ruído potencialmente adicionado aos dados, o gerenciamento das chaves, a limitação da funcionalidade em somente alguns tipos de operações básicas e a escalabilidade para que a solução continue útil em cenários de larga escala [1].

Ainda assim, as evoluções nessa área são constantes e tornam essa técnica cada vez mais útil. [25] por exemplo, criaram uma implementação de distância de Levenshtein

utilizando Criptografia Parcialmente Homomórfica, o que é de valor inclusive para a área de RE.

## 2.4 Trabalhos Relacionados

A RE é um processo essencial para consolidar informações duplicadas em panoramas críticos e que muitas vezes agregam dados distribuídos, como os sistemas de saúde. Quando se tratam de arquiteturas federadas, surgem desafios decorrentes do risco à privacidade, segurança e eficiência computacional. Partindo dessa perspectiva, o uso de criptografia homomórfica já vem sendo utilizado para assegurar transações de dados sensíveis nesse tipo de sistema.

A criptografia homomórfica também já foi utilizada em sua versão total para dar suporte à tarefa de categorização de lesões cutâneas utilizando dados do banco de imagens HAM10000, que contém fotos de lesões para treinamento de modelos de aprendizado de máquina. Para reduzir a complexidade do processo, foi implementada uma estratégia "*dropout-tolerant*" de segmentação, permitindo a transmissão de variáveis específicas das entidades, ao invés dos dados integrais. As métricas observadas incluem a precisão, que alcançou 88,43%, além de observações sobre o sistema em cenários heterogêneos, considerando características diversas dos usuários, como recursos limitados de hardware e a utilização de aprendizado federado assíncrono [19].

Outro trabalho semelhante investiga a aplicação da criptografia homomórfica total para na tarefa de classificação de arritmias cardíacas a partir de dados de eletrocardiogramas, que são convertidos em imagens 2D para análise. A criptografia homomórfica foi utilizada em conjunto com técnicas de transferência de aprendizado para garantir a eficiência do classificador. As métricas observadas incluem um F1-score de 55,13% [2].

A literatura também aborda a utilização de criptografia homomórfica parcial para a tarefa de vinculação privada de registros, onde duas partes, Alice e Bob, buscam identificar registros correspondentes em suas respectivas bases de dados sem revelar informações sensíveis. Os dados utilizados consistem em registros de bases eleitorais e de bibliotecas, que incluem informações pessoais e de produtos, respectivamente. Para reduzir a complexidade do processo de comparação, o estudo implementa funções *hash* sensíveis à localidade, que agrupam elementos semelhantes, minimizando o número de comparações necessárias. As métricas observadas incluem uma precisão de 99,3%, além de observações quanto à eficiência em termos de tempo de execução e a segurança em um modelo "semi-honesto", garantindo que a quantidade total de comparações realizadas seja mantida em sigilo [30].

As técnicas de redução de complexidade também são exploradas a fundo, e chegam a ser aplicadas mesmo nos textos cifrados. O trabalho em questão apresenta um *framework* baseado na criptografia homomórfica total para aritmética de números aproximados. A tarefa desempenhada envolve o treinamento de modelos de aprendizado de máquina, utilizando o conjunto de dados MNIST, que contém imagens de dígitos manuscritos. Para reduzir a complexidade computacional, o *framework* implementa operações de empacotamento e arredondamento homomórfico, codificando múltiplos textos plenos em uma única cifra e controlando a precisão dos resultados através de parâmetros que refletem a profundidade do circuito computacional. As métricas observadas incluem uma melhoria de pelo menos 10 vezes no tempo de treinamento, mantendo a mesma performance do modelo, com uma precisão de 96,5% para o modelo de regressão logística treinado [16].

Pesquisadores também investigaram qualitativamente a preocupação com a privacidade de dados médicos em estruturas distribuídas, e o que especialistas pensam a respeito do uso de criptografia homomórfica nesse cenário. Para abordar este problema, os autores conduziram entrevistas com *stakeholders*, utilizando “vinhetas” que exploravam o uso de criptografia homomórfica e tecnologias distribuídas. As contribuições do trabalho incluem a identificação de preocupações éticas e legais relacionadas ao uso dessas tecnologias, bem como a avaliação da disposição dos entrevistados em permitir o acesso a dados sob certas condições, como consentimento do paciente e aprovação de comitês de ética. O trabalho oferece diretrizes para a implementação responsável dessas soluções, buscando resguardar a privacidade dos pacientes em um horizonte de avanços em pesquisas na área da saúde [26].

Trabalhos mais recentes já implementam a criptografia parcialmente homomórfica em redes de *blockchain* de consórcio voltadas para a vinculação de registros, reforçando a garantia de privacidade das transações através da autenticação na rede e dos contratos inteligentes. Três conjuntos de dados públicos são utilizados, e a estratégia de bloqueio por *split bloom filters* é empregada para reduzir a complexidade da operação. Neste trabalho, é medida a qualidade da vinculação probabilística através das métricas padrão de classificadores, como a precisão, revocação e F1-Score [11].

Neste trabalho, exploramos a aplicação da criptografia homomórfica em arquiteturas federadas com foco na resolução de entidades determinística, propondo uma abordagem que visa superar as limitações de desempenho e escalabilidade identificadas nos trabalhos anteriores, além de coletar e analisar métricas referentes à qualidade da bloqueio utilizando um conjunto de dados médicos aplicável.

## 2.5 Sumário

O *framework* proposto por este trabalho visa criar e experimentar o procedimento de RE sob as condições de um sistema criptográfico homomórfico, simulando um cenário de aplicação onde as diferentes entidades de um sistema federado agem em conjunto para o incremento de uma base de dados de pacientes oncológicos. Este trabalho tem a intenção de iniciar discussões sobre os benefícios da criptografia homomórfica para a RE e suas possíveis melhorias em diversos cenários de aplicação.

## Capítulo 3

# Framework para Resolução de Entidades Homomorficamente Encriptadas em Ambientes Federados

O *framework* de RE agrega os diversos tópicos conceituados no capítulo anterior para formar uma arquitetura federada que replica um sistema colaborativo de resolução de entidades utilizando dados sintéticos inspirados em bases de pacientes oncológicos. O objetivo do *framework* é demonstrar que sua implementação permite que este procedimento seja possível sob o uso de um sistema de criptografia homomórfica, garantindo que a tarefa de resolução seja concluída sem que as cifras sejam sequer decodificadas pelo servidor. A Tabela 4.1 apresenta os atributos da base utilizada.

|    | Atributo     | Descrição                               |
|----|--------------|-----------------------------------------|
| 1  | nomep        | Primeiro nome                           |
| 2  | nomem        | Nome do meio                            |
| 3  | nomeu        | Último nome                             |
| 4  | nomemaep     | Primeiro nome da mãe                    |
| 5  | nomemaem     | Nome do meio da mãe                     |
| 6  | sexo         | Sexo                                    |
| 7  | data         | Data de nascimento                      |
| 8  | cpf          | Certificado de pessoa física brasileiro |
| 9  | cns          | ID de saúde nacional brasileiro         |
| 10 | codmunicipio | Cidade                                  |
| 11 | cep          | Código postal                           |

Tabela 3.1: Apresentação do conjunto de dados.

As seções a seguir detalham cada componente do *framework*, explicando a estratégia de blocagem utilizada, o sistema de criptografia homomórfica, aspectos sobre o ambiente de computação federado, e terminam com uma implementação exemplificada do *framework* em linguagem de programação de alto nível.

## 3.1 Estratégia de Blocagem

A estratégia de blocagem adotada é chamada de Blocagem Representativa, e se trata de um método baseado em equidade, onde um conjunto de colunas é escolhido para definir cada bloco e, para cada bloco criado, é gerada uma tupla representante deste agrupamento, populado pelos valores mais frequentes de cada uma dessas colunas. Esta estratégia foi elaborada a partir de algoritmos de RE incremental e determinística, como o FIRLA [27]. A blocagem no FIRLA é implementada através da seleção de um campo para a divisão de cada bloco, e então é feita a comparação direta dos demais campos para a segmentação de registros. Esta estratégia se mostrou eficaz tanto em cenários incrementais quanto com dados de pacientes, afinal, torna possível a identificação dessas entidades mesmo sem o codificador único (no caso deste trabalho, os campos ‘cpf’ e ‘cns’).

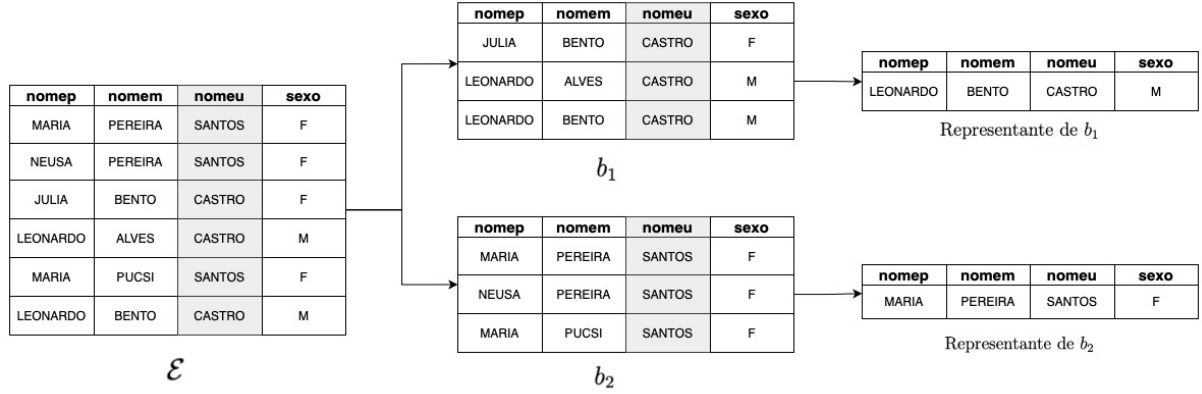
Para garantir a aplicabilidade dessa estratégia em um cenário de limitação de operações matemáticas disponíveis, simplificamos a implementação em grafos dos algoritmos mais recentes e recorremos à frequência de valores em cada campo, estratégia que se aproxima do que há de elementar em blocagem na RE, mas que ainda assim apresenta redução no custo de processamento [22].

Dessa forma, a correspondência de novos registros a blocos específicos passa a depender de uma comparação direta com essa tupla, o que torna o procedimento viável mesmo quando os dados se encontrarem criptografados, reduzindo drasticamente a quantidade de operações necessárias realizadas pelo servidor [12]. A complexidade de se construir essa tupla representante é descrita como  $O(n \times m)$ , visto que todas as colunas  $m$  de todas as tuplas  $n$  precisam ser visitadas para que o valor mais presente seja encontrado. Neste caso, a probabilidade de encontrar este valor em todas as colunas de um conjunto de dados é de  $(\frac{f_{moda}}{n})^m$ . Para valores onde a frequência da moda  $f_{moda} > 1$ , a probabilidade se torna maior do que a probabilidade de encontrar este mesmo valor elegendo uma tupla aleatória como representante  $(\frac{1}{n})^m$ . A Figura 3.1 ilustra este processo.

### 3.1.1 Métricas Fundamentais

O cálculo das métricas de qualidade da blocagem parte da suposição de que existe uma função denominada “oráculo”, capaz de resolver perfeitamente um par de registros em questão. A quantidade de comparações necessárias para a realização da Resolução de Entidades um conjunto de dados  $\mathcal{E}$  é denominada *cardinalidade*, e é denotada como  $||\mathcal{E}||$ . Tendo isso em mente, para realizar a RE em  $\mathcal{E}$ , cada tupla deve ser comparada com

Figura 3.1: Exemplificação da estratégia de Blocagem Representativa. A seleção de representantes é importante para o cenário criptografado do *framework* pois reduz a quantidade de operações criptográficas necessárias para posicionar um registro entrante.



todas as outras, ou  $||\mathcal{E}|| = |\mathcal{E}_1| \times |\mathcal{E}_2|$ . Na Resolução de Entidades Incremental, obtemos  $||\mathcal{E}||$  através da multiplicação do número de tuplas de  $\mathcal{E}$  (ou  $|\mathcal{E}|$ ) pelo número de Tuplas Entrantes  $|te|$ :

$$||\mathcal{E}|| = |\mathcal{E}| \times |te|.$$

A blocagem visa reduzir esse esforço computacional dada a sua natureza quadrática, propondo que para um conjunto de blocos  $B$  a sua cardinalidade seja definida como:

$$||\mathcal{B}|| = \sum_{b_i \in \mathcal{B}} ||b_i||,$$

onde  $||b_i||$  se trata da cardinalidade de um bloco específico  $b_i$  em relação a  $te$ . Para a Blocagem Representativa, precisamos contabilizar duas camadas de comparações, em um primeiro momento entre  $te$  e a relação de representantes  $\mathcal{R}$ , e em seguida entre a tupla entrante e os registros contidos nos blocos selecionados pela primeira camada. A cardinalidade da blocagem nesse cenário pode ser descrita como:

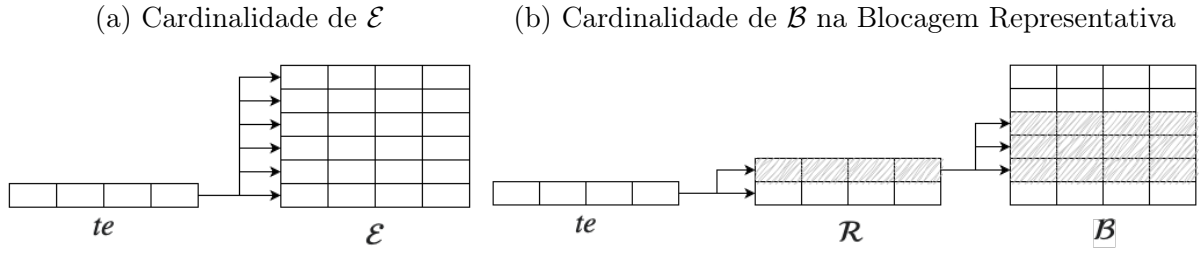
$$||\mathcal{B}|| = ||\mathcal{R}|| + \sum_{b_i \in \mathcal{R}} ||b_i||,$$

onde  $||\mathcal{B}||$  representa a cardinalidade da Blocagem Representativa,  $||\mathcal{R}||$  a cardinalidade da relação de representantes, e  $||b_i||$  passa a se tratar somente dos grupos representados por  $\mathcal{R}$  que foram selecionados pela iteração inicial (Figura 3.2) [22].

### 3.1.2 Análise da Qualidade de Blocagem

O conjunto de Réplicas Detectáveis pela função-oráculo é denotada pela função  $D$ . Para realizar a análise da qualidade da Blocagem Representativa, é necessário que as

Figura 3.2: Ilustração da redução de complexidade promovida pela Blocagem Representativa na tarefa da Resolução de Entidades Incremental.



A RE para uma  $te$  em  $\mathcal{E}$  faz uma comparação com cada registro, sendo 6 no total. Já em  $\mathcal{B}$ , a RE realiza 2 comparações de  $te$  em  $\mathcal{R}$ , mais 3 comparações em  $\mathcal{B}$ , totalizando 5 comparações.

Réplicas Detectáveis em  $\mathcal{E}$ , ou  $D(\mathcal{E})$ , sejam utilizadas como Tuplas Entrantes:

$$te = D(\mathcal{E}).$$

Assim, tanto a efetividade quanto a eficiência desse procedimento podem ser medidas e comparadas em diferentes cenários antes da submissão dos dados ao *framework* de criptografia. Considerando que uma estratégia de blocagem pode posicionar réplicas detectáveis pela função-oráculo em blocos errôneos, temos naturalmente que o conjunto de Réplicas Detectáveis na Blocagem  $D(\mathcal{B})$  é um subconjunto das Réplicas Detectáveis  $D(\mathcal{E})$ :

$$\mathcal{D}(\mathcal{B}) \subseteq \mathcal{D}(\mathcal{E}).$$

Partindo disso, três métricas de qualidade são consideradas:

1. Completude de Pares (CP), definida pela razão entre o número de réplicas detectáveis em  $\mathcal{B}$  e  $\mathcal{E}$ :

$$CP(\mathcal{B}) = |\mathcal{D}(\mathcal{B})| \div |\mathcal{D}(\mathcal{E})| \in [0, 1].$$

Em termos gerais, a CP observa o quanto uma estratégia de blocagem consegue posicionar corretamente as Réplicas Detectáveis em seus devidos blocos.

2. Qualidade de Pares (QP), definida pela razão entre o número de réplicas detectáveis em  $\mathcal{B}$  e o número total de comparações necessárias após a implementação da blocagem (ou, a cardinalidade):

$$QP(\mathcal{B}) = |\mathcal{D}(\mathcal{B})| \div ||\mathcal{B}|| \in [0, 1].$$

A QP observa uma taxa referente à quantidade de comparações necessárias para a resolução dentro de cada agrupamento resultante da blocagem.

3. Taxa de Redução (TR), que mede a redução do número de comparações em  $\mathcal{B}$  em relação à abordagem sem blocagem:

$$TR(\mathcal{B}, \mathcal{E}) = 1 - ||\mathcal{B}|| \div ||\mathcal{E}|| \in [0, 1].$$

A efetividade da blocagem está diretamente relacionada com o valor de CP, enquanto a eficiência está diretamente relacionada com os valores de QP e TR [22].

## 3.2 Resolução de Entidades Sob Criptografia Homomórfica

O *framework* proposto por este trabalho visa a demonstração da viabilidade da Resolução de Entidades Incremental em ambientes federados utilizando a criptografia homomórfica, que traz benefícios como a anonimização dos dados do lado do servidor-processador e uma gestão de chaves de criptografia simplificada. Focar no atributo incremental dessa atividade significa que partiremos de um ponto onde a RE só precise ser realizada para novos registros entrantes, o que justifica a utilização de uma arquitetura computacional de natureza colaborativa.

O *framework* implementa um esquema baseado em um sistema criptográfico parcialmente homomórfico, descrito nas seções a seguir. A escolha de um sistema parcialmente homomórfico para este trabalho se justifica pela baixa complexidade de implementação e garantia de desempenho, vitais tanto para a viabilidade da implementação quanto para sua efetividade para este trabalho [11]. Porém, seus benefícios em relação aos sistemas totalmente homomórficos são restritos a cenários de baixa complexidade de implementação. Se considerada, por exemplo, uma RE baseada em aprendizado de máquina, ou mesmo baseada em operações como similaridade de cosseno e distância de edição, sistemas totalmente homomórficos seriam mais adequados, dado o seu suporte a algoritmos mais complexos. Além disso, o uso de sistemas parcialmente homomórficos em operações mais sofisticadas muitas vezes requer passos intermediários de processamento, o que pode comprometer a garantia de privacidade destes sistemas [23]. Portanto, eventuais desdobramentos do *framework* proposto por este trabalho devem levar essa escolha em consideração para evitar os ônus citados.

### 3.2.1 Sistema Criptográfico de Paillier

As operações de RE mais avançadas tratam-se de desdobramentos de técnicas básicas de comparação entre cadeias de caracteres, ou *strings*. Nesse sentido, sistemas criptográficos homomórficos são capazes de realizar operações avançadas com *strings* [25, 24]. Porém, considerando a complexidade da implementação dessas estratégias e a limitação das operações matemáticas disponíveis na criptografia homomórfica, este trabalho concentrou-se em realizar uma tarefa básica de resolução de entidades. Assim, o conceito de que uma rede federada pode usufruir dos benefícios da criptografia homomórfica na tarefa de RE pode ser provado mesmo em soluções mais simples, que podem potencialmente evoluir.

Os Sistemas Criptográficos Parcialmente Homomórficos são capazes de realizar operações de adição em cifras, o que torna possível a comparação entre duas *strings* através da representação numérica ASCII de seus caracteres, a sua encriptação e, em seguida, uma operação de subtração homomórfica entre cada caractere cifrado de mesma posição nas duas *strings*<sup>1</sup>. Se as *strings* forem do mesmo tamanho, os resultados das subtrações são guardados em uma coleção com o mesmo tamanho das *strings* operadas, restando somente uma análise da coleção resultante para determinar se as *strings* são iguais ou não.

Com isso, observamos três cenários de resultados:

1. No primeiro, as *strings* possuem números diferentes de caracteres. Portanto, são trivialmente diferentes e não precisam ser percorridos.
2. No segundo, ilustrado na Tabela 3.2 (a), as *strings* são iguais. Portanto, o resultado da subtração homomórfica de cada um dos seus caracteres deve ser igual a 0.
3. No terceiro, ilustrado na Tabela 3.2 (b), as *strings* têm tamanhos iguais, mas são diferentes. Portanto, o resultado da subtração homomórfica de cada um dos seus caracteres apresenta números diferentes de zero.

O Sistema Criptográfico de Paillier permite a realização de operações de adição e multiplicação (não simultâneas) em números criptografados. A versão de adição deste sistema se dá pelos seguintes passos:

1. Um par de chaves é gerado, sendo uma pública e a outra privada. A primeira é utilizada na encriptação, e a segunda na descriptação.

---

<sup>1</sup>É importante lembrar que a conversão de uma *string* em sua representação numérica deve indicar o início e término de cada caractere, assim evitando o problema chamado de *colisão*, onde o resultado da conversão de letras para números não indica possibilidade da reversão dessa mesma operação.

|                       | (a)  |     |     |     |      | (b)  |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|-----------------------|------|-----|-----|-----|------|------|------|-----|------|-----|------|------|-----|-----|-----|------|-----|------|-----|-----|-----|------|
| String Original       | M    | A   | R   | I   | A    | M    | A    | R   | I    | A   | C    | L    | A   | R   | A   | L    | U   | C    | A   | S   |     |      |
|                       |      |     | ↓   |     |      |      |      | ↓   |      |     |      |      | ↓   |     |     |      |     | ↓    |     |     |     |      |
| Código ASCII          | [77, | 65, | 82, | 73, | 65]  | [77, | 65,  | 82, | 73,  | 65] | [67, | 76,  | 65, | 82, | 65] | [76, | 85, | 67,  | 65, | 83] |     |      |
|                       |      |     | ↓   |     |      |      |      | ↓   |      |     |      |      | ↓   |     |     |      |     | ↓    |     |     |     |      |
| Encriptação           | [95, | 66, | 99, | 41, | 78]* | -    | [81, | 62, | 34,  | 11, | 64]* | [89, | 56, | 71, | 43, | 82]* | -   | [33, | 17, | 27, | 95, | 11]* |
|                       |      |     |     |     | ↓    |      |      |     |      |     |      |      |     |     |     |      |     | ↓    |     |     |     |      |
| Subtração Homomórfica |      |     |     |     | [74, | 56,  | 22,  | 49, | 12]* |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     | ↓    |     |      |      |     |     |     |      |     |      | ↓   |     |     |      |
| Saída                 |      |     |     |     | [ 0, | 0,   | 0,   | 0,  | 0]   |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      | ↓   |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |
|                       |      |     |     |     |      |      |      |     |      |     |      |      |     |     |     |      |     |      |     |     |     |      |

Tabela 3.2: Subtração Homomórfica entre *strings* de tamanhos iguais. As coleções de números em vermelho (\*) indicam que os registros estão criptografados.

2. Para a encriptação de um número  $m$ , é utilizada a fórmula:

$$c = g^m \times r^n \text{ mod } n^2,$$

onde  $c$  é o número cifrado resultante,  $g$  é um gerador aleatório  $\in Z_{n^2}^*$ ,  $r$  é um número aleatório  $\in Z_{n^2}^*$  e  $n$  é o resultado da multiplicação entre dois números primos distintos  $p$  e  $q$ .

3. Para a descriptação, é necessária a chave privada  $\lambda$ , definida como o Mínimo Múltiplo Comum de  $p - 1$  e  $q - 1$ . A fórmula utilizada se dá por:

$$m = L(c^\lambda \text{ mod } n^2) \times \mu \text{ mod } n,$$

onde  $m$  é o número original,  $L(x) = \frac{x-1}{n}$  e  $\mu$  é o inverso modular de  $L(c^\lambda \times \text{mod } n^2) \text{ mod } n$ .

4. Para efetuar a soma entre dois números  $c_1$  e  $c_2$  cifrados a partir de duas entradas  $m_1$  e  $m_2$ , basta realizar a multiplicação modular desses termos:

$$c = c_1 \times c_2 \text{ mod } n^2.$$

Uma vez descriptado o valor de  $c$ , obtém-se o resultado da soma entre os termos  $m_1$  e  $m_2$ .

Para atender ao caso de uso deste trabalho, é necessário garantir que esse sistema criptográfico também seja capaz de realizar a subtração entre duas cifras. Temos que essa operação representa a operação *inversa* do que se deseja quando se tratam de dois termos não cifrados. Portanto, para que o sistema seja capaz de realizar a subtração de termos criptografados, basta que a operação realizada no contexto cifrado também seja invertida. Fazemos isso através da fórmula:

$$c = c_1 \times (c_2^{-1} \text{ mod } n^2) \text{ mod } n^2,$$

onde  $c$  se trata do resultado cifrado da subtração entre dois termos  $c_1$  e  $c_2$ , obtido através da multiplicação do primeiro com o inverso modular do segundo [28].

O sistema de Paillier nos permite realizar a operação de subtração entre dois números criptografados, o que também possibilita que duas *strings* sejam comparadas. Essa é a premissa da técnica mais elementar da RE, utilizada na prova de conceito que propomos com a sua aplicação em ambientes federados.

### 3.2.2 Análise da Qualidade de Resolução de Entidades

Levando em consideração que a corretude do sistema criptográfico de Paillier é garantida pelo Teorema de Fermat-Euler (Apêndice B) – *i.e.*, o conteúdo cifrado é integralmente recuperado por operações de decodificação –, um caso de testes para a RE é elaborado. Neste cenário, é possível verificar a qualidade da operação atômica de RE utilizada no *framework*, ao eleger pares de tuplas com chaves primárias idênticas como representantes da mesma entidade. Dessa forma, este campo é desconsiderado e a qualidade da classificação é testada comparando os demais atributos do conjunto ao observar as seguintes métricas:

1. Coeficiente de Jaccard, que mede a sobreposição entre o conjunto de valores preditos como positivo e aqueles que de fato são positivos:

$$\text{Coeficiente de Jaccard} = \frac{VP}{VP + FP + FN},$$

onde  $VP$  é o número de Verdadeiros positivos,  $FP$  o de Falsos Positivos e  $FN$  Falsos Negativos.

2. Precisão, responsável por calcular a proporção de pares de entidades corretamente resolvidos em relação ao total de pares de entidades que a técnica identificou como correspondentes:

$$\text{Precisão} = \frac{VP}{VP + FP}.$$

3. Revocação (ou *Recall*), usada para medir a proporção de pares de entidades verdadeiramente correspondentes que foram identificados pela técnica, em relação ao total de pares que deveriam ter sido identificados:

$$\text{Revocação} = \frac{VP}{VP + FN}.$$

4. F1-Score, que é a média harmônica entre a precisão e a revocação, oferecendo um equilíbrio entre as duas métricas.:

$$F_1 = 2 \cdot \frac{Precisão \cdot Revocação}{Precisão + Revocação}.$$

Dessa forma, é possível avaliar aspectos importantes da operação de RE, não só como a sua capacidade de tomar decisões corretas, mas também como é a qualidade de suas classificações, considerando circunstâncias onde falsos positivos ou falsos negativos apresentam mais criticidade.

### 3.3 Ambiente de Computação Federado

A arquitetura de computação federada considerada neste trabalho consiste na interação entre duas entidades: o Servidor, que centraliza os dados e realiza a operação de RE; e o Cliente, que em um estado inicial pode enviar uma base integral para o servidor, ou consultar em uma base já existente se a sua tupla possui réplicas representantes de uma mesma entidade ou não [9]. Estes papéis são descritos nas seções a seguir.

#### 3.3.1 Servidor de Processamento Centralizado

O Servidor representa a unidade de processamento centralizada na rede federada. Nele são armazenados dois conjuntos de dados, a Tabela Integral e a Relação de Representantes. A Tabela Integral se trata da versão criptografada dos dados originais com seu ID não criptografado, mais uma coluna não criptografada contendo o ID de seu representante, o que automaticamente indica a qual bloco o registro pertence. A Relação de Representantes contém as tuplas representantes de cada bloco, contendo um ID único não criptografado e as mesmas colunas da tabela original, porém criptografadas. As tuplas representantes são obtidas através da composição de valores que representam aquele de maior frequência dentro de sua coluna em específico.

O servidor também armazena a chave criptográfica pública utilizada no sistema, tornando-o assim capaz de criptografar dados, mas não de descriptografá-los.

#### Requisitos Funcionais de Processamento Computacional

- O Servidor deve ser capaz de receber uma tupla entrante criptografada e retornar a Relação de Representantes já subtraída homomorficamente.

- O Servidor deve ser capaz de receber uma tupla entrante criptografada juntamente com o ID de seu bloco representante, e retornar um subconjunto da Tabela Integral contendo os registros vinculados ao bloco representante subtraídos homomorficamente.
- O Servidor deve ser capaz de realizar a operação homomórfica propriamente dita, que recebe como parâmetro a tupla entrante e o subconjunto dos dados a ser iterado. O código ASCII encriptado de cada célula do conjunto de dados é substituído pelo resultado de sua subtração homomórfica realizada com a tupla entrante, mantendo assim a estrutura relacional entre ID do registro e coluna.

Nesse método, o subconjunto é percorrido tupla a tupla e, em seguida, juntamente com a tupla entrante, também são percorridos coluna a coluna. As células varridas por essa iteração são inicialmente verificadas pela quantidade de caracteres. Se o número for diferente, a diferença entre as strings é trivial e então a célula é anulada para o somatório final. Se o número de caracteres for igual, o método percorre cada uma das letras criptografadas, subtraindo homomorficamente o seu valor e incluindo o resultado na estrutura de resposta.

### 3.3.2 Clientes Colaboradores

A entidade Cliente representa as unidades computacionais colaboradoras da rede federada. Nele são armazenadas a chave pública de encriptação e também a chave privada de desencriptação. É a partir de um dos clientes que o estado inicial da aplicação é criado.

#### Requisitos Funcionais de Processamento Computacional

- O Cliente que contiver a Tabela Integral deve ser capaz de encriptá-la utilizando a chave pública de criptografia e enviá-la ao Servidor.
- O Cliente deve ser capaz de eleger os representantes (ou realizar a blocagem) a partir do conjunto de dados da Tabela Integral e do conjunto de colunas escolhido para a blocagem.
- O Cliente deve ser capaz de realizar a desencriptação dos dados, recebendo uma coleção de dados e utiliza a chave privada para transformar os dados cifrados recebidos do servidor em texto legível de novo.
- O Cliente deve ser capaz de computar os valores encontrados nesses conjuntos de dados, dando uma pontuação específica para cada tupla relacionada a quantas de

suas colunas são iguais às colunas da tupla entrante. Essa verificação é realizada iterando ao longo das listas de resposta obtidas do processamento feito pelo servidor, descritografando-as e verificando se o valor da subtração das strings é igual a 0. Se isso ocorrer, então as palavras comparadas são iguais e é somado um ponto à pontuação de resolução atribuída ao registro.

## 3.4 Implementação

O fluxo do processo de Resolução de Entidades utilizando Criptografia Homomórfica busca representar as interações entre clientes e servidor de uma rede computacional federada. Para isso, utiliza uma instância da entidade Servidor chamada *Alice* e duas instâncias da entidade Cliente chamadas *Bob* e *Carol*. Neste exemplo, *Bob* é responsável pela criação do estado inicial do sistema, executando a estratégia de bloqueio e fornecendo os dados encriptados para o servidor. *Carol* representa um segundo colaborador da rede que, mesmo não tendo acesso aos dados no estágio inicial onde *Bob* alimentou o sistema, ainda sim é capaz de realizar a resolução de sua entidade através do uso da chave pública compartilhada. Neste esquema, o servidor *Alice* fica incumbido de armazenar e realizar a operação homomórfica de subtração nos dados, não tendo acesso em momento algum a eles em seu formato não encriptado. Os conjuntos de dados (*datasets*) envolvidos no processo são pontuados na Tabela 3.3.

| Descrição                 | Dataset | Versão Encriptada | Versão Pontuada | Encriptada e Pontuada |
|---------------------------|---------|-------------------|-----------------|-----------------------|
| Tabela Integral           | ti      | eti               | sti             | esti                  |
| Relação de Representantes | rr      | err               | srr             | esrr                  |
| Tupla Entrante            | te      | ete               | -               | -                     |

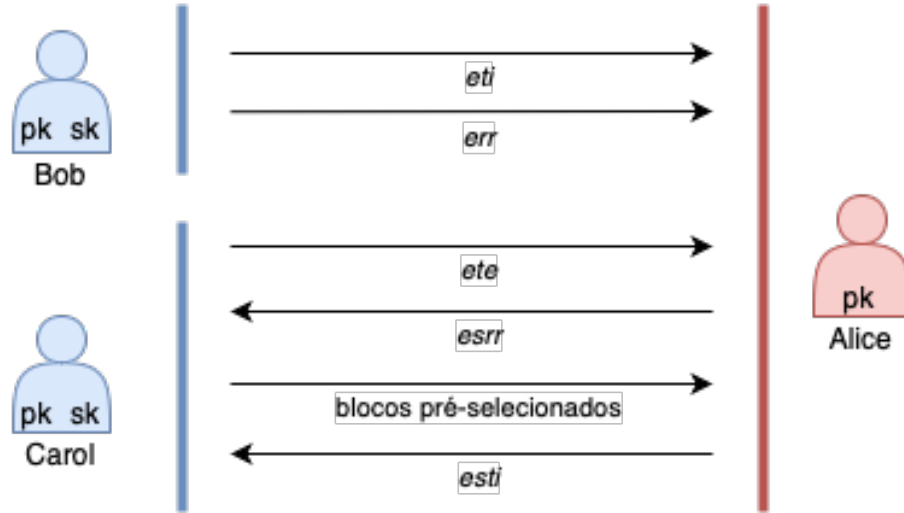
Tabela 3.3: Conjuntos de dados alocados pelo *framework* em tempo de execução.

O processo e as interações entre os componentes descritos nas seções anteriores são ilustrados na Figura 3.3. Observando a figura, nota-se que os dados são sempre trafegados sob criptografia, sendo o prefixo “e” o denominador de variáveis encriptadas que contêm cada conjunto de dados representado.

O cliente *Bob* representa a consolidação da base de consulta em ambiente compartilhado, já *Carol* representa o estágio incremental do *framework*, consultando via o servidor *Alice* e de maneira encriptada eventuais registros duplicados em relação à sua Tupla Entrante *te*. O trabalho se concentra nestes dois nós da rede, representando os tipos possíveis de interação pretendidos. Adicionar múltiplos nós a essa rede significaria

o aumento de dados disponíveis para consulta via servidor, e também das operações de consulta. Porém, a complexidade computacional de ambas as operações se manteria na mesma ordem de  $O(n)$ , como descrito na seção 2.1.2.

Figura 3.3: Transações entre cliente e servidor do *framework*.



### 3.4.1 Código-fonte

A implementação do Framework de Resolução de Entidades proposto foi realizada utilizando a linguagem Python 3.12.2. A premissa do paradigma de Programação Orientada a Objetos de representar entidades do mundo real foi utilizada para replicar o cenário de um sistema de contribuição federado de Resolução de Entidades para a base de dados utilizada no trabalho. Os Algoritmos 3.1 e 3.2 propõem classes que atendem aos requisitos funcionais apontados na seção 3.3.

### 3.4.2 Exemplo de Implementação

Uma vez definidas as classes, é necessário seguir os passos ilustrados anteriormente na Figura 3.3 para realizar a RE através do *framework*. A seguir, utilizamos os dados de exemplo ilustrados na Figura 3.1 para ilustrar a implementação do *framework*.

---

```

1  Classe Cliente:
2      Funcao Cliente(pk, sk): # Construtor
3          Cliente.pk = pk
4          Cliente.sk = sk
5      Funcao elege_representantes(ti, colunas):
6          rr = copia estrutura ti
7          grupos = agrupa ti por colunas
8          ti['bloco'] = 0 # inicializa coluna 'bloco' vazia
9
10         b = 0
11         Para cada grupo em grupos:
12             condicoes = (ti[col] == grupo[col] &) para cada col em colunas
13             filtrados = filtra ti por condicoes
14             ti[filtrados.indices, 'bloco'] = b
15             b++
16
17             representante = {} # inicializa um registro vazio
18             Para cada col em filtrados.colunas:
19                 representante[col] = agrupa filtrados por col, seleciona
20                     valor mais frequente
21             insere representante em rr
22     Retornar ti, rr
23 Funcao encriptar(dados):
24     dados_enc = copia dados
25     Para cada registro em dados_enc:
26         Para cada coluna em registro:
27             int_val = [] # inicializa vetor ASCII vazio
28             Para cada caractere em registro[coluna]:
29                 insere Cliente.pk.encriptar(ASCII(caractere)) em int_val
30             dados_enc[registro.indice, coluna] = int_val
31     Retornar dados_enc
32 Funcao escore(es):
33     s = copia es
34     s = Cliente.desencriptar(s)
35     s['pontoss'] = 0 # inicializa coluna 'bloco' vazia
36     Para cada registro em s:
37         Para cada coluna em registro:
38             Se todos os numeros de registro[coluna] == 0:
39                 s[registro.indice, 'pontoss']++
40     Retornar s
41 Funcao desencriptar(es):
42     s = copia es
43     Para cada registro em es:
44         Para cada coluna em registro:
45             s[registro.indice, coluna] +=
46                 Cliente.sk.desencripta(caractere) para cada caractere em
47                 registro[coluna]
48     Retornar s

```

---

Algoritmo 3.1: Pseudocódigo com a implementação da classe Cliente.

---

```

1  Classe Servidor:
2      Funcao Servidor(eti, err, pk): # Construtor
3          Servidor.eti = eti
4          Servidor.err = err
5          Servidor.pk = pk
6
7      Funcao subtracao_homomorfica(ete, blocos):
8          Se blocos != Nulo: # Verifica se o segundo parametro foi informado
9              dados = filtra Servidor.eti por blocos
10         Senao:
11             dados = Servidor.err
12
13         Para cada registro em bloco_dados:
14             Para cada coluna em registro:
15                 Se tamanho(registro[coluna]) != tamanho(ete[coluna]):
16                     dados[registro.indice, coluna] = Nulo
17                     Continue
18
19                 dados[registro.indice, coluna] = [] # inicializa lista vazia
20                 i = 0
21                 Para cada valor em registro[coluna]:
22                     sub = (valor - ete[coluna][i]) # subtracao homomorfica
23                     insere sub em dados[registro.indice, coluna]
24                     i++
25         Retornar dados

```

---

Algoritmo 3.2: Pseudocódigo com a implementação da classe Servidor.

### 3.4.2.1 Instância do cliente Bob e alocação de dados na memória

As chaves privada *sk* e pública *pk* são instanciadas a partir de uma implementação externa do Sistema Criptográfico de Paillier [7], e a elas são atribuídos os métodos con-  
dizentes com as operações disponíveis para cada uma (i.e., *pk* possui o método *encriptar* e *sk* possui o método *desencriptar*). Discussões sobre a implementação desse sistema criptográfico e a sua prova de corretude estão presentes nos Apêndices A e B.

*ti* é então carregado, contendo o conjunto de dados de pacientes oncológicos já preparados, limpando linhas com registros vazios (Tabela 3.4).

*Bob* é instanciado a partir da classe Cliente. O método *Bob.elege\_representantes* é usado para criar *rr* e atualizar *ti* com a coluna de blocos indicando a qual representante seus registros pertencem (Algoritmo 3.3). A Tabela 3.5 mostra os dados contidos nas variáveis *ti* e *rr*, agora incluindo a coluna “bloco”, criada através do algoritmo 3.3.

| <i>ti</i> |  | nomep    | nomem   | nomeu  | nomemaep  | nomemaem | sexo | ... | cep      |
|-----------|--|----------|---------|--------|-----------|----------|------|-----|----------|
| 3         |  | MARIA    | PEREIRA | SANTOS | MARLUSE   | ASSUNCAO | F    | ... | 36660000 |
| 31        |  | NEUSA    | PEREIRA | SANTOS | LUIZA     | JESUS    | F    | ... | 36660000 |
| 45        |  | JULIA    | BENTO   | CASTRO | MARIA     | RIBEIRO  | F    | ... | 32450000 |
| 47        |  | LEONARDO | ALVES   | CASTRO | APARESIDA | CASTRO   | M    | ... | 35570000 |
| 87        |  | MARIA    | PUCSI   | SANTOS | MARIA     | BORJES   | F    | ... | 37002140 |
| 117       |  | LEONARDO | BENTO   | CASTRO | MIRTES    | NETO     | M    | ... | 36700000 |

Tabela 3.4: Amostra do conjunto de dados *ti*.

---

```

1 Bob = Cliente(pk, sk)
2 colunas = ['nomeu']
3 ti, rr = Bob.elege_representantes(ti, colunas)

```

---

Algoritmo 3.3: Instanciação do cliente Bob e chamada ao método de eleição de representantes.

| <i>ti</i> |  | bloco | nomep    | nomem   | nomeu  | nomemaep  | nomemaem | sexo | ... | cep      |
|-----------|--|-------|----------|---------|--------|-----------|----------|------|-----|----------|
| 3         |  | 1     | MARIA    | PEREIRA | SANTOS | MARLUSE   | ASSUNCAO | F    | ... | 36660000 |
| 31        |  | 1     | NEUSA    | PEREIRA | SANTOS | LUIZA     | JESUS    | F    | ... | 36660000 |
| 45        |  | 0     | JULIA    | BENTO   | CASTRO | MARIA     | RIBEIRO  | F    | ... | 32450000 |
| 47        |  | 0     | LEONARDO | ALVES   | CASTRO | APARESIDA | CASTRO   | M    | ... | 35570000 |
| 87        |  | 1     | MARIA    | PUCSI   | SANTOS | MARIA     | BORJES   | F    | ... | 37002140 |
| 117       |  | 0     | LEONARDO | BENTO   | CASTRO | MIRTES    | NETO     | M    | ... | 36700000 |

| <i>rr</i> |  | bloco | nomep    | nomem   | nomeu  | nomemaep  | nomemaem | sexo | ... | cep      |
|-----------|--|-------|----------|---------|--------|-----------|----------|------|-----|----------|
| 0         |  | 0     | LEONARDO | BENTO   | CASTRO | APARESIDA | CASTRO   | M    | ... | 32450000 |
| 1         |  | 1     | MARIA    | PEREIRA | SANTOS | LUIZA     | ASSUNCAO | F    | ... | 36660000 |

Tabela 3.5: Conjuntos de dados *ti* e *rr*, resultantes do processo de eleição de representantes do Algoritmo 3.3.

### 3.4.2.2 Encriptação de *ti* e *rr*

Neste passo, *Bob* utiliza o método *Bob.encryptar* para encriptar os dados tanto de *ti* quanto de *rr* (Algoritmo 3.4). Cada registro se torna uma lista de números encriptados, correspondentes ao código ASCII de cada caractere do valor original. A Tabela 3.6 mostra o conteúdo de *eti* e *err*, versões cifradas de *ti* e *rr*. O tamanho das cifras é definido pelo quadrado do tamanho da chave pública.

---

```

1 eti = Bob.encryptar(ti)
2 err = Bob.encryptar(rr)

```

---

Algoritmo 3.4: Os conjuntos *ti* e *rr* passam pelo processo de encriptação utilizando o sistema de Paillier.

| <i>eti</i> |              |              |              |              |                 |                 |             |            |            |
|------------|--------------|--------------|--------------|--------------|-----------------|-----------------|-------------|------------|------------|
|            | <b>bloco</b> | <b>nomep</b> | <b>nomem</b> | <b>nomeu</b> | <b>nomemaep</b> | <b>nomemaem</b> | <b>sexo</b> | <b>...</b> | <b>cep</b> |
| <b>3</b>   | 1            | [10987452,   | [12738947,   | [19203847,   | [72938415,      | [39847102,      | [16384792,  | ...        | [52739104, |
| <b>31</b>  | 1            | [38719487,   | [97412039,   | [34712093,   | [28374109,      | [84729106,      | [19283745,  | ...        | [83649271, |
| <b>45</b>  | 0            | [29831749,   | [38479120,   | [10293847,   | [64291783,      | [28374915,      | [28491736,  | ...        | [41928375, |
| <b>47</b>  | 0            | [91028746,   | [49381093,   | [65827394,   | [19482734,      | [10928374,      | [23847109,  | ...        | [75832049, |
| <b>87</b>  | 1            | [32897187,   | [12783948,   | [12398471,   | [76049218,      | [93741028,      | [98374912,  | ...        | [64389172, |
| <b>117</b> | 0            | [81098374,   | [38471928,   | [91827364,   | [51837492,      | [72938461,      | [91029384,  | ...        | [29584716, |

| <i>err</i> |              |              |              |              |                 |                 |             |            |            |
|------------|--------------|--------------|--------------|--------------|-----------------|-----------------|-------------|------------|------------|
|            | <b>bloco</b> | <b>nomep</b> | <b>nomem</b> | <b>nomeu</b> | <b>nomemaep</b> | <b>nomemaem</b> | <b>sexo</b> | <b>...</b> | <b>cep</b> |
| <b>0</b>   | 0            | [17283945,   | [68529473,   | [84927163,   | [23489167,      | [96847123,      | [42938517,  | ...        | [68247193, |
| <b>1</b>   | 1            | [90418273,   | [31247859,   | [57632841,   | [71029384,      | [83756192,      | [19384725,  | ...        | [54937218, |

Tabela 3.6: Conjuntos de dados *eti* e *err*, resultantes do processo de encriptação do Algoritmo 3.4. Os sinais de '[' e ',' denotam uma coleção de cifras geradas aleatoriamente, semelhantes à primeira descrita pelo valor da célula.

### 3.4.2.3 Criação do ponto inicial de contribuição

Neste ponto, ao criar e encriptar *ti* e *rr*, *Bob* se torna responsável pela criação do ponto inicial de contribuição da rede federada. Instancia-se então *Alice* a partir da classe Servidor. Nela são armazenados *eti*, *err* e *pk*, impossibilitando assim que o servidor tenha acesso aos dados descriptados, ou mesmo que seja capaz de descriptá-los, uma vez que *sk* não foi compartilhado com *Alice* (Algoritmo 3.5).

---

```
1 Alice = Servidor(eti, err, pk)
```

---

Algoritmo 3.5: Instanciação do servidor Alice. *sk* não é compartilhada com o servidor, o que garante que qualquer dado compartilhado com Alice seja incapaz de ser descriptado.

### 3.4.2.4 Preparação da Tupla Entrante

Com os dados já carregados no servidor, o segundo cliente *Carol* instanciado a partir da classe Cliente deseja resolver uma entidade *te*. As chaves utilizadas para a criação de *Carol* são as mesmas utilizadas na criação de *Bob*, indicando que tanto a chave pública quanto a privada foram compartilhadas entre os clientes da rede federada. Uma vez instanciado, o cliente utiliza o método *encriptar* para transformar *te* em sua versão encriptada *ete* antes de enviá-lo ao servidor (Algoritmo 3.6, Tabela 3.7).

---

```

1      Carol = Cliente(pk, sk)
2      ete = Carol.encryptar(te)

```

---

Algoritmo 3.6: Instanciação do cliente Carol e encriptação da tupla entrante *te*.

| <i>te</i> |          |       |          |          |          |      |     |          |
|-----------|----------|-------|----------|----------|----------|------|-----|----------|
|           | nomep    | nomem | nomeu    | nomemaep | nomemaem | sexo | ... | cep      |
| 0         | VANDENIR | SILVA | ASSUNCAO | MARLUSE  | ASSUNCAO | F    | ... | 36660000 |

| <i>ete</i> |            |            |            |            |            |            |     |            |
|------------|------------|------------|------------|------------|------------|------------|-----|------------|
|            | nomep      | nomem      | nomeu      | nomemaep   | nomemaem   | sexo       | ... | cep        |
| 0          | [84321679, | [95713428, | [68294317, | [73184529, | [46927835, | [52173986, | ... | [37498216, |

Tabela 3.7: *ete* é criado a partir da encriptação da tupla entrante *te*.

### 3.4.2.5 Subtração homomórfica e pontuação da Relação de Representantes

*rr*

*ete* é então enviado para o servidor *Alice* para que a subtração homomórfica seja realizada com a relação de representantes encriptada *err*. O resultado da operação é retornado em *esrr*, contendo em cada célula uma lista com o resultado de cada subtração homomórfica feita entre os campos da tupla entrante *ete* e *err* caso o número de caracteres seja igual. Caso contrário, o valor da célula é anulado (Algoritmo 3.7, Tabela 3.8).

---

```

1      esrr = Alice.subtracao_homomorfica(ete)

```

---

Algoritmo 3.7: Subtração homomórfica do conjunto de dados realizada entre *err* e *ete* pelo servidor Alice, resultando em *esrr*.

| esrr |       |            |            |            |            |          |            |     |            |
|------|-------|------------|------------|------------|------------|----------|------------|-----|------------|
|      | bloco | nomep      | nomem      | nomeu      | nomemaep   | nomemaem | sexo       | ... | cep        |
| 0    | 0     | Nulo       | Nulo       | [87194623, | Nulo       | Nulo     | [25674319, | ... | [93871542, |
| 1    | 1     | [84273195, | [19345687, | [75982134, | [68423957, | Nulo     | [31579284, | ... | [49283651, |

Tabela 3.8: Amostra do conjunto de dados cifrados *esrr*, criado a partir da subtração homomórfica do Algoritmo 3.7.

Neste ponto, *Carol* recebe *esrr* e realiza a sua descriptação e escoragem. O resultado é exibido em *srr*, contendo as mesmas listas e valores nulos de *esrr*, agora descriptados. A coluna ‘pontos’ contém um número inteiro referente à quantidade de colunas que tiveram os resultados da subtração homomórfica igual a 0 para todos os caracteres, indicando que não há diferença entre os valores comparados (Algoritmo 3.8, Tabela 3.9).

---

```
1 srr = Carol.escore(esrr)
```

---

Algoritmo 3.8: Escoragem realizada pelo cliente *Carol* a partir de *esrr*.

| srr |       |        |                         |                          |                          |                 |          |     |                              |
|-----|-------|--------|-------------------------|--------------------------|--------------------------|-----------------|----------|-----|------------------------------|
|     | bloco | pontos | nomep                   | nomem                    | nomeu                    | nomemaep        | nomemaem | ... | cep                          |
| 0   | 0     | 1      | Nulo                    | Nulo                     | [-16, 0, 5,<br>0, 3, -4] | Nulo            | Nulo     | ... | [4, 5, 0, 9,<br>-9, 0, 1, 2] |
| 1   | 1     | 8      | [-1, -4, -3,<br>-10, 0] | [0, 0, 0, 0,<br>0, 0, 0] | [0, 0, 0, 0,<br>0, 0]    | [0, 0, 0, 0, 0] | Nulo     | ... | [0, 0, 0, 0,<br>0, 0, 0, 0]  |

Tabela 3.9: Amostra do conjunto de dados *srr*, resultante da escoragem do algoritmo 3.8. As colunas ‘pontos’ e ‘bloco’ mostram que o bloco 1 contabilizou 8 colunas de valores iguais à tupla entrante *te*, sendo o bloco selecionado pela sua pontuação mais alta.

#### 3.4.2.6 Subtração homomórfica e pontuação da Tabela Integral *ti*

Os blocos de maior pontuação são selecionados para varredura. O processo de subtração homomórfica se repete, agora comparando a tupla entrante encriptada *ete* com os registros do bloco de maior pontuação. O resultado da operação é retornado em *esti*, contendo em cada célula uma lista com o resultado de cada subtração homomórfica feita entre os campos da tupla entrante *ete* e *eti* caso o número de caracteres seja igual. Caso contrário, o valor da célula é anulado (Algoritmo 3.9, Tabela 3.9).

---

```
1 blocos_max = [1]
2 esti = Alice.subtracao_homomorfica(ete, blocos_max)
```

---

Algoritmo 3.9: Chamada à subtração homomórfica realizada entre *eti* e *ete*, resultando em *esti*.

| esti |       |            |            |            |            |            |     |            |
|------|-------|------------|------------|------------|------------|------------|-----|------------|
|      | bloco | nomep      | nomem      | nomeu      | nomemaep   | nomemaem   | ... | cep        |
| 3    | 1     | [16283947, | [92713485, | [34827196, | Nulo       | Nulo       | ... | [94682173, |
| 31   | 1     | [75429318, | [61342978, | [57219364, | [23496571, | [75981324, | ... | [31298457, |
| 87   | 1     | [48923657, | Nulo       | [84162735, | [68347219, | Nulo       | ... | [56789234, |

Tabela 3.10: Amostra do conjunto de dados **cifrados** *esti*, resultante da subtração homomórfica do Algoritmo 3.9.

A partir de *esti*, *Carol* realiza a sua descriptação e escoragem. O resultado é exibido em *sti*, contendo as mesmas listas e valores nulos de *esti*, agora descriptados. A coluna ‘pontos’ contém um número inteiro referente à quantidade de colunas que tiveram os resultados da subtração homomórfica igual a 0 para todos os caracteres, indicando que não há diferença entre os valores comparados (Algoritmo 3.10, Tabela 3.11).

---

```
1 sti = Carol.escore(est)
```

---

Algoritmo 3.10: Escoragem realizada pelo cliente *Carol* a partir de *esti*.

| sti       | bloco | pontos | nomep                   | nomem                    | nomeu                 | nomemaep               | nomemaem        | ... | cep                         |
|-----------|-------|--------|-------------------------|--------------------------|-----------------------|------------------------|-----------------|-----|-----------------------------|
| <b>3</b>  | 1     | 5      | [-1, -4, -3,<br>-10, 0] | [0, 0, 0, 0,<br>0, 0, 0] | [0, 0, 0, 0,<br>0, 0] | Nulo                   | Nulo            | ... | [0, 0, 0, 0, 0, 0,<br>0, 0] |
| <b>31</b> | 1     | 10     | [0, 0, 0, 0,<br>0]      | [0, 0, 0, 0,<br>0, 0, 0] | [0, 0, 0, 0,<br>0, 0] | [0, 0, 0, 0, 0]        | [0, 0, 0, 0, 0] | ... | [0, 0, 0, 0, 0, 0,<br>0, 0] |
| <b>87</b> | 1     | 2      | [-1, -4, -3,<br>-10, 0] | Nulo                     | [0, 0, 0, 0,<br>0, 0] | [1, -20, 9, -17,<br>0] | Nulo            | ... | [0, 0, 0, 0, 0, 0,<br>0, 0] |

Tabela 3.11: Amostra da escoragem realizada pelo Algoritmo 3.10. A coluna ‘pontos’ mostra que a tupla de índice 31 contabilizou 10 colunas de valores iguais à tupla entrante *te*, sendo o registro que satisfaz o requisito da função-oráculo para a resolução de entidades.

A resolução de entidades é concluída utilizando a função-oráculo. Para este exemplo, a tupla de índice 31 de *ti* armazenada no servidor *Alice* contém 10 das 11 colunas de valor equivalente à tupla entrante *te* enviada por *Carol*, indicando que ambas se referem à mesma entidade. O Capítulo 4 a seguir mostra os resultados experimentais observados com a aplicação deste *framework*.

## Capítulo 4

# Resultados Experimentais

Os resultados experimentais consistem em medições a respeito da blocagem e da resolução de entidades propostas. Os procedimentos foram aplicados em um conjunto de dados fictício, baseado em dados demográficos de pacientes oncológicos.

### 4.1 Caracterização dos Dados

A estrutura de dados foi escolhida para ilustrar a criticidade dos contextos em que a RE está inserida. Originalmente, a base possuía 400.000 registros. Após a remoção das tuplas com dados faltantes, restaram 11.327 delas. Outros procedimentos de limpeza mais simples foram utilizados para melhorar a legibilidade das informações e normalizar os formatos dos dados. A Tabela 4.1 apresenta os atributos da base utilizada.

|    | Atributo     | Descrição                               |
|----|--------------|-----------------------------------------|
| 1  | nomep        | Primeiro nome                           |
| 2  | nomem        | Nome do meio                            |
| 3  | nomeu        | Último nome                             |
| 4  | nomemaep     | Primeiro nome da mãe                    |
| 5  | nomemaem     | Nome do meio da mãe                     |
| 6  | sexo         | Sexo                                    |
| 7  | data         | Data de nascimento                      |
| 8  | cpf          | Certificado de pessoa física brasileiro |
| 9  | cns          | ID de saúde nacional brasileiro         |
| 10 | codmunicipio | Cidade                                  |
| 11 | cep          | Código postal                           |

Tabela 4.1: Apresentação do conjunto de dados.

O dataset tratado, com 11.327 registros, conta com 11.066 nomes únicos (combinação de ‘nomep’, ‘nomem’ e ‘nomeu’), além de 261 homônimos (registros que possuem esses três campos simultaneamente repetidos em um ou mais outros registros). Em rela-

ção ao nome da mãe, são 6.335 nomes únicos e 4.991 homônimos, considerando os campos ‘nomemaep’ e ‘nomemaem’. A base abrange pacientes de um total de 746 municípios, sendo aproximadamente 59,23% desses pacientes do sexo feminino. A Tabela 4.2 mostra exemplos de registros após a preparação.

|        | nomep     | nomem            | ... | cpf         | cns             | codmunicipio         | cep      |
|--------|-----------|------------------|-----|-------------|-----------------|----------------------|----------|
| 3      | TEREZINHA | FERREIRA         | ... | 33685053604 | 107953622820018 | GUANHAES_MG          | 35740000 |
| 31     | VANDENIR  | SILVA            | ... | 7151872798  | 898001176669739 | ALEM PARAIBA_MG      | 36660000 |
| 45     | MARIA     | EFLJENIA RIBEIRO | ... | 68354312615 | 801434168311037 | BELO HORIZONTE_MG    | 32450000 |
| 47     | CARLOS    | ROBERTO          | ... | 451013611   | 898050011070791 | FORMIGA_MG           | 35570000 |
| 87     | SUELI     | APARESIDA COSTA  | ... | 6984474680  | 898001174534232 | VARGINHA_MG          | 37002140 |
| ...    | ...       | ...              | ... | ...         | ...             | ...                  | ...      |
| 399848 | EFLJENIA  | PEDRO A          | ... | 63580594672 | 801434155005870 | ITABIRITO_MG         | 35450000 |
| 399874 | JOANA     | DARC             | ... | 4337739602  | 898050042249904 | UBERLANDIA_MG        | 38405382 |
| 399920 | SEBASTIAO | GUIDO ALVES      | ... | 71213627672 | 801434159556219 | SANTANA DA VARGEM_MG | 37195000 |
| 399932 | DANIEL    | PAULA            | ... | 3350145698  | 898050051341001 | UBERLANDIA_MG        | 38405382 |
| 399959 | JOSUE     | MENDES           | ... | 24071609672 | 898001166658025 | ITUIUTABA_MG         | 38302042 |

Tabela 4.2: Dados oncológicos sintetizados utilizados no trabalho.

## 4.2 Blocagem Representativa

Utilizamos o conjunto de dados oncológicos de 11.327 registros para medir a qualidade da estratégia de blocagem proposta. Para isso, iniciamos definindo a função-oráculo, capaz de detectar réplicas com exatidão. Para o contexto deste trabalho, consideramos que dois registros representam a mesma entidade se apresentarem 10 valores em comum entre as 11 colunas totais do conjunto de dados integral <sup>1</sup>. A partir disso, selecionamos as tuplas que atendem a este requisito. Ao todo, 32 registros apresentam réplicas detectáveis pela função-oráculo definida anteriormente, ou  $|\mathcal{D}(\mathcal{E})| = 32$ . Medimos então a qualidade da blocagem em diferentes cenários.

### 4.2.1 Cenários de Blocagem por Conjuntos de Colunas

Selecionamos a coluna “nomeu” para a blocagem. Agrupando os registros, obtemos 1860 blocos. Com essa blocagem, mais de 60% dos registros foram alocados no grupo de *nomeu* igual a “SILVA”, e apresentaram uma média de 1,2 registros por grupo. Cada

<sup>1</sup>Este valor é arbitrário. Por definição, a Resolução de Entidades implica que a tarefa deve ser realizada entre duas tuplas *não-idênticas*, o que do contrário se trataria de um problema de Deduplicação.

um destes blocos tem relacionada a eles uma tupla com os valores mais comuns de cada coluna dentro deste bloco, nomeada a sua Tupla Representante. Por exemplo, a blocagem por “nomeu” resulta, entre outros, no grupo “ARAUJO”, de 100 registros. A Tupla Representante deste grupo é mostrada na Tabela 4.3.

| Atributo     | Valor             |
|--------------|-------------------|
| nomep        | MARIA             |
| nomem        | ALVES             |
| nomeu        | ARAUJO            |
| nomemaep     | MARIA             |
| nomemaem     | ARAUJO            |
| sexo         | F                 |
| data         | 1960-09-05        |
| cpf          | 1011304651        |
| cns          | 103904124620007   |
| codmunicipio | BELO HORIZONTE_MG |
| cep          | 38405382          |

Tabela 4.3: Representante do grupo “ARAUJO”, no cenário de blocagem por “nomeu”. O valor das colunas representa o valor mais recorrente da coluna no conjunto de dados original.

As Tuplas Entrantes  $te$  (aqui as Réplicas Detectáveis  $\mathcal{D}(\mathcal{E})$ , como descrito na seção 3.1.2) são comparadas com cada um dos representantes dos grupos, e aquelas com maior quantidade de valores de coluna em comum são selecionados para a Resolução de Entidades com os registros originais atrelados a estes grupos. Aplicando este procedimento ao cenário de blocagem por “nomeu”, posicionamos cada uma das réplicas detectáveis de  $\mathcal{D}(\mathcal{E})$  em um grupo. Ao final, temos que 24 das 32 réplicas detectáveis de  $\mathcal{D}(\mathcal{E})$  foram posicionadas corretamente em seu grupo correspondente, ou  $|\mathcal{D}(\mathcal{B})| = 24$ .

Partindo dos valores obtidos de  $|\mathcal{D}(\mathcal{E})|$  e  $|\mathcal{D}(\mathcal{B})|$ , calculamos a Completude de Pares dessa estratégia:

$$CP(\mathcal{B}) = |\mathcal{D}(\mathcal{B})| \div |\mathcal{D}(\mathcal{E})| = 75\%.$$

Para definir a cardinalidade referente à blocagem  $||\mathcal{B}||$ , precisamos medir quantas comparações são necessárias para bloquear cada uma das  $\mathcal{D}(\mathcal{B})$ , considerando que só podem ser feitas comparações com elementos de um mesmo bloco. Sendo assim, precisamos contabilizar o número de comparações feitas entre as  $te$  com a Relação de Representantes (ou  $||\mathcal{R}||$ ) e somar este resultado à quantidade de comparações feitas entre estas  $te$  e as tuplas de cada bloco de maior afinidade entre um  $te$  e seu representante (ou  $\sum_{bi \in \mathcal{R}} ||b_i||$ ).

Para o cenário de blocagem “nomeu”, temos que o número de blocos  $|\mathcal{B}| = 1.860$ . Multiplicamos este número pelo número de Réplicas Detectáveis  $||\mathcal{R}|| = |\mathcal{B}| \times |\mathcal{D}(\mathcal{E})| = 59.520$ . Com o número de comparações com representantes de blocos  $||\mathcal{R}||$  obtido, somamos a ele o número de comparações feitas entre os blocos eleitos para obter  $||\mathcal{B}||$ . No caso

da blocagem por “nomeu”, 64 blocos diferentes foram percorridos. Considerando inclusive que diferentes registros podem ter consultado o mesmo bloco, 4.395 novas comparações devem ser feitas entre as réplicas detectáveis e as tuplas originais dos blocos selecionados. Chegamos então ao número  $||\mathcal{B}|| = ||\mathcal{R}|| + \sum_{bi \in \mathcal{R}} ||b_i|| = 59.520 + 4.395 = 63.915$ .

Podemos então calcular a Qualidade de Pares para esse cenário de blocagem:

$$QP(\mathcal{B}) = |\mathcal{D}(\mathcal{B})| \div ||\mathcal{B}|| = 24 \div 63.915 \approx 0,037549\%.$$

A partir do número de registros do conjunto de dados integral e o número de Réplicas Detectáveis, obtemos a sua cardinalidade  $||\mathcal{E}|| = 11.327 \times 32 = 362.464$ , e podemos então calcular a Taxa de Redução:

$$TR(\mathcal{B}, \mathcal{E}) = 1 - ||\mathcal{B}|| \div ||\mathcal{E}|| = 1 - 63.915 \div 362.464 \approx 82,36\%.$$

Repetimos este processo para diferentes cenários de escolhas de colunas, e obtemos as métricas destacadas nas Tabelas 4.4 e 4.5.

| Colunas      | Blocos | $CP(\mathcal{B})$ | $QP(\mathcal{B})$ | $TR(\mathcal{B}, \mathcal{E})$ |
|--------------|--------|-------------------|-------------------|--------------------------------|
| nomep        | 2.880  | 87,50%            | 0,028873%         | 73,24%                         |
| nomem        | 4.656  | 81,25%            | 0,017366%         | 58,69%                         |
| nomeu        | 1.860  | 75,00%            | 0,037549%         | 82,36%                         |
| nomemaep     | 2.061  | 81,25%            | 0,035526%         | 79,80%                         |
| nomemaem     | 1.704  | 78,12%            | 0,038630%         | 82,14%                         |
| sexo         | 2      | 100,00%           | 0,016872%         | 47,67%                         |
| data         | 9.199  | 100,00%           | 0,010867%         | 18,76%                         |
| cpf          | 11.243 | 0,00%             | 0,000000%         | 0,72%                          |
| cns          | 11.265 | 0,00%             | 0,000000%         | 0,52%                          |
| codmunicipio | 746    | 71,87%            | 0,074392%         | 91,40%                         |
| cep          | 4.937  | 0,00%             | 0,000000%         | 56,02%                         |

Tabela 4.4: Métricas de qualidade da Blocagem Representativa para agrupamentos de 1 coluna.

Observando as métricas assinaladas na Tabela 4.4, notamos que a Blocagem Representativa alcançou números expressivos para a Taxa de Redução  $TR(\mathcal{B}, \mathcal{E})$ , chegando a apresentar 91,4% na blocagem por “codmunicipio”. Do contrário, a Qualidade de Pares  $QP(\mathcal{B})$  se manteve abaixo de 0,1% para todas as escolhas de coluna. A Completude de Pares  $CP(\mathcal{B})$  apresentou números elevados de maneira majoritária, mas também apresentou porcentagens nulas em 3 dos casos estudados.

A tabela 4.5 mostra as combinações de 2 colunas que apresentaram uma Taxa de Redução acima de 50%. Assim, conseguimos filtrar, das 55 combinações possíveis, as 7 que conseguiram reduzir mais a complexidade da RE<sup>2</sup>. Observando as métricas, percebemos

<sup>2</sup>As 165 diferentes possibilidades de blocagem utilizando 3 colunas também foram analisadas, mas nenhuma apresentou uma Taxa de Redução acima de 50%. Como a tendência desses agrupamentos é se tornar cada vez mais específicos com o aumento do número de blocos, os agrupamentos de 4 ou mais colunas não foram realizados. Todas as análises estão disponíveis com o código-fonte do trabalho.

| Colunas            | Blocos | $CP(\mathcal{B})$ | $QP(\mathcal{B})$ | $TR(\mathcal{B}, \mathcal{E})$ |
|--------------------|--------|-------------------|-------------------|--------------------------------|
| nomep, sexo        | 2.949  | 87,50%            | 0,028235%         | 72,64%                         |
| nomem, sexo        | 5.019  | 90,62%            | 0,017985%         | 55,51%                         |
| nomeu, sexo        | 2.268  | 87,50%            | 0,036834%         | 79,02%                         |
| nomemaep, sexo     | 2.639  | 87,50%            | 0,031291%         | 75,31%                         |
| nomemaem, sexo     | 2.104  | 93,75%            | 0,040249%         | 79,43%                         |
| sexo, codmunicipio | 1.257  | 90,62%            | 0,063318%         | 87,36%                         |
| codmunicipio, cep  | 5.294  | 0,00%             | 0,000000%         | 52,89%                         |

Tabela 4.5: Métricas de qualidade da Blocagem Representativa para agrupamentos de 2 colunas.

que o mesmo comportamento dos agrupamentos de uma coluna é apresentado: números expressivos de  $TR(\mathcal{B}, \mathcal{E})$ , números baixos de  $QP(\mathcal{B})$ , e  $CP(\mathcal{B})$  majoritariamente alto, mas com ocorrências nulas. É pouco conclusivo também que uma ou outra configuração se relacione consistentemente com as métricas de  $CP(\mathcal{B})$  e  $QP(\mathcal{B})$ , visto que a arbitrariedade da configuração do agrupamento e a amostra de resposta observada influenciam diretamente em  $D(\mathcal{B})$ , numerador de ambas as métricas. A arbitrariedade da estratégia de blocagem representativa também afeta  $TR(\mathcal{B}, \mathcal{E})$ , dada a presença de  $|D(\mathcal{E})|$  em sua fórmula. Isso explica porque casos como a blocagem por ‘sexo’ (2 blocos e 47,67% de  $TR(\mathcal{B}, \mathcal{E})$ ) apresenta  $TR(\mathcal{B}, \mathcal{E})$  menor que na blocagem por ‘codmunicipio’ (746 blocos e 91,40% de  $TR(\mathcal{B}, \mathcal{E})$ ): blocos não selecionados via tupla representante não são percorridos e, portanto, não aumentam a cardinalidade da blocagem  $||\mathcal{B}||$ , numerador de  $TR(\mathcal{B}, \mathcal{E})$ .

Em termos de efetividade e eficiência, os valores mostram que a Blocagem Representativa é eficaz para a maioria das escolhas de coluna, dados os números de  $CP(\mathcal{B})$ . Porém, ela é inconsistente em termos de eficácia, visto que, por um lado o número de comparações cai drasticamente, como observado pelos altos números de  $TR(\mathcal{B}, \mathcal{E})$ , mas a grande maioria das comparações realizadas não revela uma Réplica Detectável (i.e.,  $QP(\mathcal{B})$  se mantém baixo). Considerando o contexto ao qual foi proposta a Blocagem Representativa, números altos de  $TR(\mathcal{B}, \mathcal{E})$  significam não só que menos comparações são necessárias, mas também significam que menos operações homomórficas complexas foram necessárias, o que pode indicar que a premissa da Blocagem Representativa é aderente ao *framework* federado de Resolução de Entidades.

### 4.2.2 Variação do Critério de Resolução

O critério utilizado para comparar as métricas de blocagem nas seções anteriores considera que dois registros se referem à mesma entidade se 10 de 11 colunas possuem

valores idênticos. Variar este valor, em teoria, significa alterar a qualidade da classificação em detrimento do número de comparações necessárias para resolver entidades. A Tabela 4.6 mostra as métricas de qualidade de algumas das blocagens por coluna única ao variar o número do critério de resolução<sup>3</sup>.

| Blocagem     | Métrica                        | 5 de 11 campos | 6 de 11   | 7 de 11   | 8 de 11   | 9 de 11   |
|--------------|--------------------------------|----------------|-----------|-----------|-----------|-----------|
| nomep        | $CP(\mathcal{B})$              | 68,57%         | 77,11%    | 90,13%    | 91,27%    | 88,54%    |
|              | $QP(\mathcal{B})$              | 0,021089%      | 0,024501% | 0,029761% | 0,030247% | 0,029275% |
|              | $TR(\mathcal{B}, \mathcal{E})$ | 71,29%         | 72,21%    | 73,26%    | 73,36%    | 73,30%    |
| nomeu        | $CP(\mathcal{B})$              | 75,67%         | 77,46%    | 80,26%    | 80,16%    | 77,08%    |
|              | $QP(\mathcal{B})$              | 0,032663%      | 0,035040% | 0,039180% | 0,038772% | 0,036918% |
|              | $TR(\mathcal{B}, \mathcal{E})$ | 79,55%         | 80,48%    | 81,91%    | 81,75%    | 81,57%    |
| nomemaep     | $CP(\mathcal{B})$              | 51,02%         | 65,85%    | 78,95%    | 83,33%    | 84,38%    |
|              | $QP(\mathcal{B})$              | 0,016986%      | 0,023089% | 0,030137% | 0,031798% | 0,030985% |
|              | $TR(\mathcal{B}, \mathcal{E})$ | 73,48%         | 74,82%    | 76,87%    | 76,86%    | 75,96%    |
| sexo         | $CP(\mathcal{B})$              | 100%           | 100%      | 100%      | 100%      | 100%      |
|              | $QP(\mathcal{B})$              | 0,016811%      | 0,017160% | 0,017070% | 0,016861% | 0,016749% |
|              | $TR(\mathcal{B}, \mathcal{E})$ | 47,48%         | 48,55%    | 48,28%    | 47,64%    | 47,29%    |
| cns          | $CP(\mathcal{B})$              | 0%             | 0%        | 0%        | 0%        | 0%        |
|              | $QP(\mathcal{B})$              | 0%             | 0%        | 0%        | 0%        | 0%        |
|              | $TR(\mathcal{B}, \mathcal{E})$ | 0,54%          | 0,53%     | 0,53%     | 0,53%     | 0,53%     |
| codmunicipio | $CP(\mathcal{B})$              | 60,87%         | 69,01%    | 73,68%    | 78,57%    | 79,17%    |
|              | $QP(\mathcal{B})$              | 0,045508%      | 0,059270% | 0,067666% | 0,070789% | 0,067787% |
|              | $TR(\mathcal{B}, \mathcal{E})$ | 88,19%         | 89,72%    | 90,39%    | 90,20%    | 89,69%    |

Tabela 4.6: Métricas de qualidade da Blocagem Representativa para diferentes valores do critério de resolução. Cada coluna de valores numéricos representa a quantidade mínima de atributos idênticos para considerar um par de registros como representantes de uma mesma entidade, variando de 5 a 9 atributos.

<sup>3</sup>As métricas das demais blocagens também foram quantificadas, mas foram ocultadas da relação por apresentarem uma tendência semelhante às presentes. O código-fonte com a análise completa foi disponibilizada no repositório do trabalho [17].

As Figuras 4.1, 4.2 e 4.3 ilustram a tendência das métricas de  $CP(\mathcal{B})$ ,  $QP(\mathcal{B})$  e  $TR(\mathcal{B}, \mathcal{E})$ , respectivamente, ao variar o número de atributos do critério de resolução entre 5 e 10.

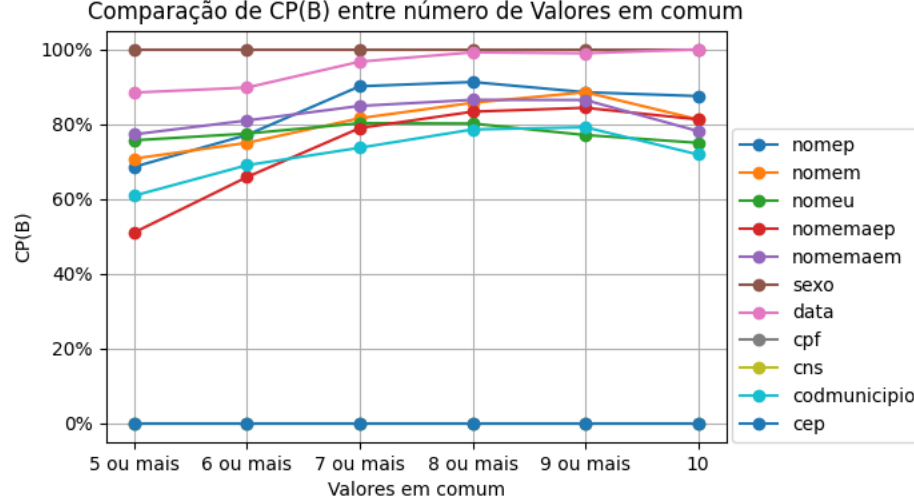


Figura 4.1: Variação da métrica de Completude de Pares  $CP(\mathcal{B})$  entre diferentes números de atributos do critério de resolução.

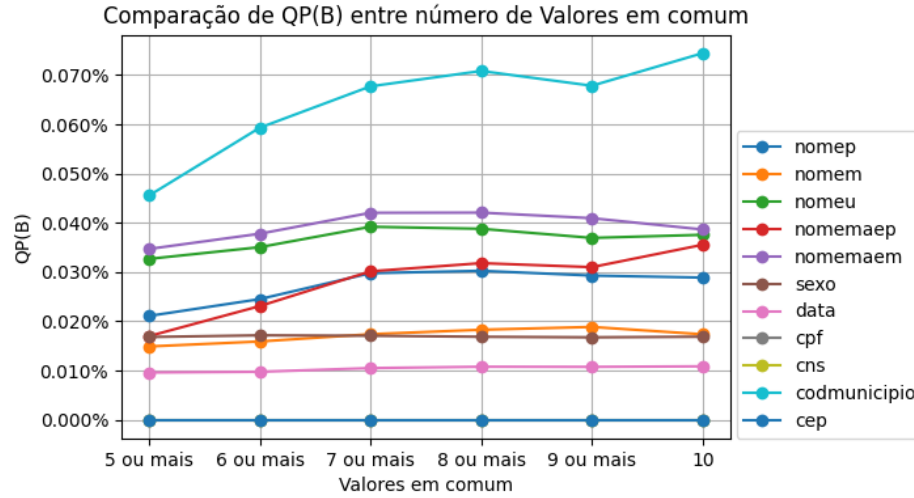


Figura 4.2: Variação da métrica de Qualidade de Pares  $QP(\mathcal{B})$  entre diferentes números de atributos do critério de resolução.

Observando a tabela e as figuras, é possível observar alguns pontos. Primeiro, a configuração tanto da função-oráculo quanto da estratégia de blocagem influenciam nas métricas de qualidade do classificador. Por exemplo, em termos de  $QP(\mathcal{B})$ , a estratégia de blocagem por ‘codmunicipio’ se destaca das demais. Por outro lado, para ambas as métricas de  $CP(\mathcal{B})$  e  $QP(\mathcal{B})$ , a estratégia de blocagem por ‘cep’ apresentou valores nulos, o que indica que, para todas as configurações da função-oráculo testadas, todas tiveram dificuldade em classificar corretamente as réplicas detectáveis.

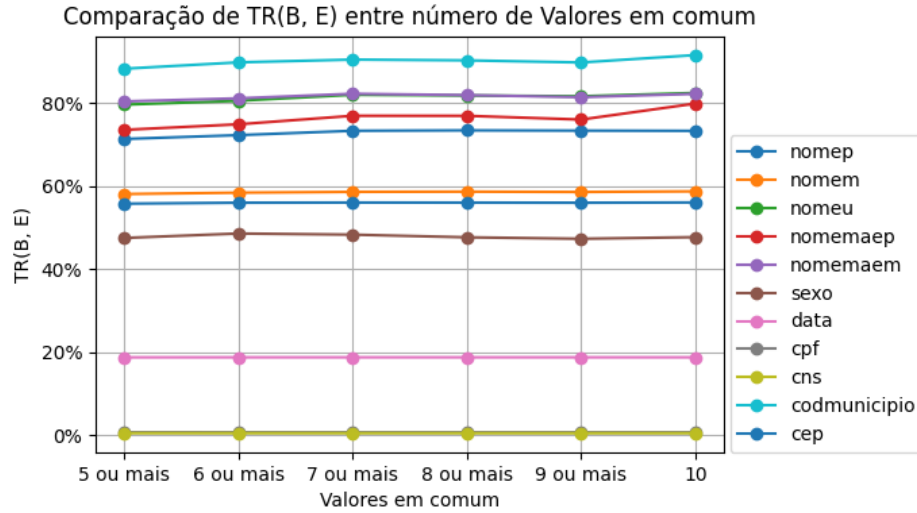


Figura 4.3: Variação da métrica de Taxa de Redução  $TR(\mathcal{B}, \mathcal{E})$  entre diferentes números de atributos do critério de resolução.

Outra observação diz respeito ao comportamento dos gráficos com as alterações de configuração. É esperado que a  $TR(\mathcal{B}, \mathcal{E})$  se mantenha estável independente do número de campos escolhido como limiar de similaridade, uma vez que, tanto o denominador quando o numerador da fórmula são multiplicados pelo número de réplicas detectáveis  $D(\mathcal{E})$ , o que mantém a proporcionalidade. As oscilações são pequenas, e acontecem por causa da distribuição de números diferentes de registros entre os blocos, bem como pela quantidade de blocos criada.

Para as métricas de  $CP(\mathcal{B})$  e  $QP(\mathcal{B})$ , existe também um comportamento esperado, que é da ascensão relacionada ao aumento do limiar de similaridade. É intuitivo pensar que, até certo ponto, quanto menos campos em comum forem considerados para classificar um par de tuplas como réplica, menos sensível é o classificador. Como o critério de classificação é usado inclusive para a confecção dos blocos, fica claro que a qualidade dos blocos decai, e mais registros são classificados de maneira equivocada.

Porém, há dois comportamentos que inclinam sua motivação ao domínio do problema. O primeiro, de uma certa “estabilidade” ao utilizarmos as 7 ou mais valores em comum como limiar. Isso indica que motivos como a correlação entre dois ou mais campos pode tornar a comparação entre estes valores com as tuplas entrantes desnecessária (por exemplo, ‘cep’ e ‘codmunicipio’, ou um nome usualmente restrito a um gênero específico, como “MARIA” e “F”).

Outro comportamento é o decaimento da métrica em pontos destoantes de limiares mais altos, como nitidamente acontece na linha de ‘codmunicipio’, do gráfico das métricas de  $QP(\mathcal{B})$  (Figura 4.2). Aqui acontece uma desproporção entre o número de réplicas detectáveis  $|\mathcal{D}(\mathcal{B})|$  e o número de comparações necessárias para a RE dentro da blocagem adotada  $\|\mathcal{R}\|$  e  $\sum_{bi \in \mathcal{R}} \|b_i\|$ . Na prática, os componentes da fórmula se comportam de maneira intuitiva: tanto  $|\mathcal{D}(\mathcal{B})|$  quanto  $\|\mathcal{B}\|$  decaem com o aumento do limiar, ou, quanto

mais exigente o classificador, menos réplicas serão classificadas e menos comparações serão necessárias para realizar essa tarefa. Porém, é contra-intuitivo, mas a proporção entre  $|\mathcal{D}(\mathcal{B})|$  e  $||\mathcal{B}||$  não necessariamente segue esse comportamento, uma vez que  $|\mathcal{D}(\mathcal{B})|$  e  $||\mathcal{B}||$  são grandezas independentes, e isso influencia diretamente a métrica de  $QP(\mathcal{B})$ <sup>4</sup>. A Tabela 4.7 ilustra esse fenômeno.

| Métrica                                           | 8 de 11   | 9 de 11   | 10 de 11   |
|---------------------------------------------------|-----------|-----------|------------|
| $ \mathcal{D}(\mathcal{B}) $                      | 99        | 76        | 23         |
| $  \mathcal{B}  $                                 | 139.852   | 112.116   | 30.917     |
| $ \mathcal{D}(\mathcal{B})  \div   \mathcal{B}  $ | 0,0010532 | 0,0010612 | 0,00096347 |

Tabela 4.7: Detalhamento das métricas componentes de  $QP(\mathcal{B})$ . Tanto  $|\mathcal{D}(\mathcal{B})|$  quanto  $||\mathcal{B}||$  decaem com o aumento do limiar de campos, porém a proporção entre essas duas métricas não se mantém constante.

Em termos de eficiência e eficácia, além da configuração de blocagem, o ajuste do limiar da função-oráculo também pode ajudar a melhorar o classificador. Cabe ao usuário analisar o contexto de aplicação e entender até onde a calibragem desse limiar traz benefícios, em detrimento do aumento de operações computacionais e necessidade de aperfeiçoamento do algoritmo de classificação.

### 4.3 Resolução de Entidades

A operação de RE proposta pelo trabalho acontece através de comparações entre os caracteres de cada campo entre pares de registros. Para atestar a sua eficácia, as tuplas com valores de “cpf” em comum foram consideradas como réplicas detectáveis. O campo foi escolhido por se tratar de um identificador único de um paciente. A partir disso, o campo foi retirado do conjunto original e a RE foi realizada com os campos restantes. O critério de resolução utilizado para a função-oráculo foi de 7 de 11 campos idênticos, configuração que apresentou maior F1-Score. A Figura 4.4 mostra os valores de Verdadeiros Positivos  $VP$ , Falsos Negativos  $FN$ , Falsos Positivos  $FP$  e Verdadeiros Negativos  $VN$  obtidos.

Caracterizando estes 4 indicadores para o contexto deste trabalho, os  $VP$  significam que registros oncológicos que de fato contêm uma réplica detectável, conseguiram ser detectados pelo classificador automático. Da mesma forma, os  $VN$  mostram o nú-

<sup>4</sup>O mesmo fenômeno acontece para a métrica de  $CP(\mathcal{B})$ , onde componentes se comportam de maneira esperada, mas ocorre um decaimento entre suas proporções para números mais altos do limiar de classificação.

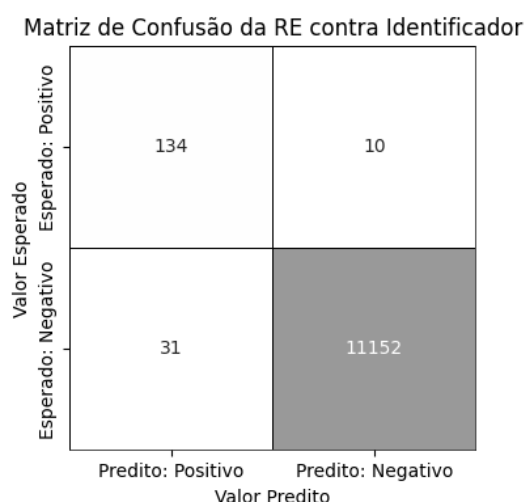


Figura 4.4: Matriz de Confusão da RE com 7 de 11 campos como critério de resolução. O quadrante mais escuro reforça a presença majoritária de registros sem réplica e que assim foram classificados pelo algoritmo de RE.

mero de registros oncológicos que não possuíam réplicas detectáveis foram corretamente classificadas como tal.

Por outro lado, os *FN* indicam quantos registros que possuíam uma réplica detectável foram ignorados. Intuitivamente, espera-se que este número seja de fato maior que zero, dada a baixa sofisticação da técnica usada na classificação. Este ponto é, inclusive, abordado pela literatura como o problema da otimização dos classificadores, tornando-os cada vez mais capaz de não serem enganados por registros que de fato se assemelham muito, mas que no final não se referem à mesma entidade. Por fim, o número de *FP* se mantém o menor – e de fato este comportamento é intuitivo para esta configuração. *FP* neste caso significa que uma tupla que não deveria ser identificada como réplica, foi classificada como tal. Na prática, isso significa que duas tuplas coincidem em 7 dos 11 campos diferentes, mas possuem o identificador único ‘cpf’ diferente entre si<sup>5</sup>.

### 4.3.1 Coeficiente de Jaccard

A partir destes números, é calculado o Coeficiente de Jaccard:

<sup>5</sup>Aqui, atestamos a influência da arbitrariedade das configurações do classificador: em um cenário onde o campo de desambiguação escolhido foi o ‘cns’, o que em tese é o identificador único do paciente no sistema nacional de saúde, o número de *FP* chega a ultrapassar o número de *FN*. Especificamente neste caso, isso acontece porque pode haver a criação de um novo ‘cns’ para o mesmo paciente caso ele se mude de cidade, alterando somente 3 dos 11 campos (‘cns’, ‘codmunicipio’ e ‘cep’), e, portanto, se mantendo dentro do critério de detecção estabelecido.

$$\text{Coeficiente de Jaccard} = \frac{134}{134 + 10 + 31} \approx 76,57\%$$

Este número aponta que o classificador conseguiu identificar corretamente 76,57% de pacientes que possuem réplicas na base de dados através do critério da função-oráculo, em relação ao número de pacientes cujas réplicas foram identificadas através do campo ‘cpf’. Este coeficiente mede a discrepância entre a interseção e a união das réplicas identificadas, fornecendo uma visão mais resumida e global sobre o classificador. Essa métrica é especialmente importante para se obter uma visão introdutória de um algoritmo de classificação binário onde os registros são majoritariamente rotulados como negativos.

### 4.3.2 Precisão

A precisão do classificador é calculada em seguida:

$$\text{Precisão} = \frac{134}{134 + 10} = 93,05\%$$

A precisão avalia a qualidade das classificações positivas feitas pelo classificador, que aqui apresenta 93,05% de acertos. É importante observar a precisão quando os falsos positivos são críticos no contexto do problema. No cenário simulado por este trabalho, por exemplo, números altos de falsos positivos podem significar o vínculo de prontuários de pacientes distintos, levando a diagnósticos equivocados.

### 4.3.3 Revocação

A próxima métrica tomada é a revocação:

$$\text{Revocação} = \frac{134}{134 + 31} \approx 81,21\%$$

Essa métrica indica que, de todos os casos onde os pacientes deveriam ser considerados como réplicas, o modelo conseguiu identificar corretamente 81,21%. A revocação é especialmente importante em casos onde os falsos negativos são mais impactantes. Ao se tratar de pacientes oncológicos, por exemplo, falhar em resolver uma entidade pode significar manter o prontuário de um paciente fragmentado, impedindo a rastreabilidade e influenciando no seu tratamento.

A relação entre a revocação e a precisão obtidos com a estratégia de classificação adotada é ajustada pela configuração do número de campos utilizado na função-oráculo. Isso quer dizer que diminuir o número de campos necessários para classificar réplicas significa aumentar a revocação (mais registros serão classificados como réplicas), mas diminuir a precisão (a chance de acerto é menor pois a similaridade também é menor), e vice-versa.

#### 4.3.4 F1-Score

O F1-Score é a próxima métrica observada:

$$F_1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}} \approx 86,73\%$$

O F1-Score harmoniza a precisão e a revocação. Aumentar o valor do F1-Score da RE significa maximizar o número de réplicas detectadas pelo classificador ao passo de minimizar a quantidade de classificações totais (inclusive as classificações erradas) ele precisou realizar. Dentre as configurações testadas, utilizar 7 campos na função-oráculo apresentou o maior F1-Score, de 86,73%, seguido por 77,32% utilizando 8 campos, 67,41% com 6 campos e 39,02% com 9 campos.

#### 4.3.5 Desempenho Computacional

A última análise realizada por este trabalho é uma observação quanto ao desempenho computacional da resolução de entidades sob as diretrizes propostas pelo *framework*. A máquina utilizada para a execução dos experimentos é um MacBook Air M2 2022, com 8 núcleos e 16GB de memória. A operação atômica de RE consiste na conversão da cadeia de caracteres de cada campo em um vetor de códigos ASCII, subtraindo cada elemento com os elementos dos vetores da tupla entrante, caso os vetores possuam o mesmo tamanho.

São comparados quatro cenários de implementação em termos de tempo de execução para a tarefa de RE com uma tupla entrante:

1. **Sem bloqueio e sem criptografia homomórfica (SBSC):** Tempo necessário para realizar a RE comparando o número mínimo de campos propostos de uma tupla com todos os registros do conjunto.

2. **Com blocagem e sem criptografia homomórfica (CBSC):** Tempo necessário para realizar a RE comparando uma tupla com os registros da estratégia de blocagem<sup>6</sup>.
3. **Sem blocagem e com criptografia homomórfica (SBCC):** Tempo necessário para realizar a RE comparando uma tupla com todos os registros do conjunto, considerando que os dados já estão criptografados.
4. **Com blocagem e com criptografia homomórfica (CBCC):** Tempo necessário para realizar a RE comparando uma tupla com os registros da estratégia de blocagem, considerando que os dados já estão criptografados.

A Figura 4.5 relaciona os cenários descritos com a variação do critério de resolução.

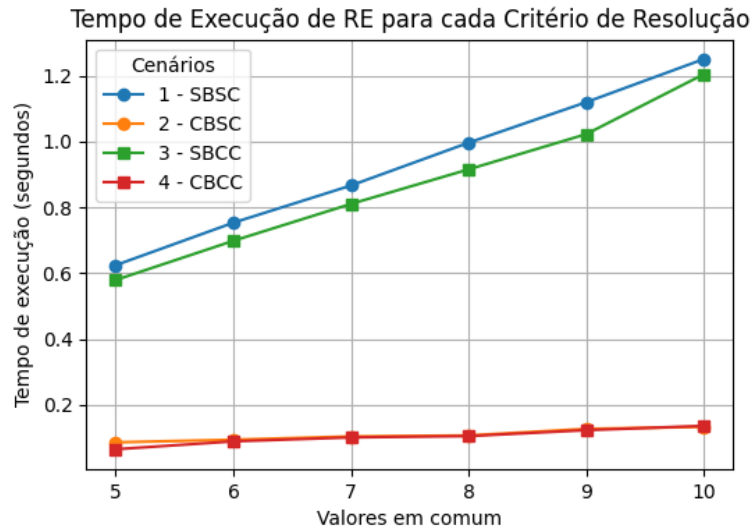


Figura 4.5: Tempo de execução em segundos para uma operação de RE em função do número de valores em comum, para os quatro cenários descritos.

Através do gráfico, é possível observar que os cenários 2 e 4, onde há a aplicação da blocagem, apresentam os menores tempos de execução para a RE, enquanto os cenários 1 e 3, onde não há blocagem, concentram os maiores tempos de execução. Além disso, o gráfico mostra que a criptografia pouco interfere no tempo de execução da RE, mantendo a versão criptografada dos cenários semelhante (no caso do cenário 3, até menor) a sua versão sem criptografia.

Ao analisar estes dados, atestamos a utilidade do *framework*. O procedimento garante a anonimização dos dados trafegados em redes federadas e pouco influencia no custo computacional da operação, sendo ainda otimizada pela possibilidade de aplicação de estratégias de blocagem que reduzem o número total de operações ( $TR(\mathcal{B}, \mathcal{E})$ ) buscando comprometer minimamente a qualidade do resultado final ( $CP(\mathcal{B})$ ,  $QP(\mathcal{B})$ ).

<sup>6</sup>A estratégia escolhida para esta análise foi por “codmunicípio”, por apresentar a maior métrica de  $TR(\mathcal{B}, \mathcal{E})$ , ainda assim mantendo valores significativos de  $CP(\mathcal{B})$  e  $QP(\mathcal{B})$ .

## Capítulo 5

# Conclusão e Trabalhos Futuros

Este trabalho buscou experimentar a perspectiva de que sistemas de Criptografia Homomórfica viabilizam a realização da tarefa de Resolução de Entidades em arquiteturas federadas de maneira a garantir a segurança do tráfego e armazenamento de dados, comprometendo ao mínimo o desempenho computacional dos componentes envolvidos neste processo. Ao sugerir uma estratégia de blocagem que prioriza a Taxa de Redução da classificação, concretizar a sua implementação utilizando o sistema criptográfico de Paillier representa a possibilidade de implementação de estratégias mais otimizadas em ambientes consolidados para resolver problemas mais complexos.

Implementado em Python, um protótipo do *framework* proposto aplica a estratégia de Blocagem Representativa que reduz em até cerca de 90% a cardinalidade das comparações necessárias para realizar a Resolução de Entidades em um conjunto de dados criado com o intuito de representar dados de pacientes oncológicos. Respalhada pela correteude do algoritmo criptográfico de Paillier, a estratégia de blocagem apresenta números significativos de Completude de Pares a depender da escolha de colunas, o que significa que a solução como um todo seria capaz de entregar valor à tarefa de RE federada com certo grau de efetividade e eficiência, mesmo sob uma camada adicional de segurança.

Aspectos a respeito do procedimento para a RE também são analisados. Ao variar o critério utilizado pela função-oráculo – que consiste na escolha do número de colunas mínimas semelhantes entre dois registros para que sejam consolidados – é possível observar que há influência sobre as métricas de blocagem, apresentando um declínio de  $CP(\mathcal{B})$  e  $QP(\mathcal{B})$  para os números mais baixos de valores em comum, mas estabilizando ao se aproximar dos 8 ou mais campos. O teste da RE contra um campo de identificação único apresenta um Coeficiente de Jaccard mediano, mas evolui em termos de F1-Score. De maneira geral, é importante ressaltar que tais observações ressaltam a validade do *framework* como arcabouço arquitetural, incentivando a variação e experimentação com técnicas alternativas tanto de blocagem, quanto de RE, ou até mesmo de sistema criptográfico homomórfico.

O caráter pioneiro da pesquisa abre oportunidades de evolução em diversos âmbitos envolvidos no trabalho. Tratando-se da tarefa de Resolução de Entidades, a evolução da técnica de comparação entre cadeias de caracteres pode significar um aumento de com-

plexidade acentuado no sistema federado como um todo, dada a limitação das operações disponíveis na criptografia homomórfica. Isso também pode acontecer com as técnicas de agrupamento e blocagem, que atualmente chegam a empregar algoritmos de aprendizado de máquina para a realização do agrupamento dimensional das tuplas. Incrementar a complexidade da técnica de blocagem pode acarretar inclusive no problema de reagrupamento após a introdução de novas tuplas, o que, se designado ao Servidor de um sistema federado, este precisaria realizar essa tarefa com os registros ainda criptografados.

Quanto à estratégia de blocagem, o trabalho conseguiu propor uma estratégia de blocagem aderente à premissa do sistema: um agrupamento de simples implementação, que viabiliza a sua reprodução utilizando as operações básicas da criptografia homomórfica, mas que ainda assim apresenta métricas altas de redução de custo computacional. Porém, ainda há muito a ser explorado. Estudos futuros podem realizar a comparação entre diferentes estratégias de blocagem homomórfica nos mesmos termos de efetividade e eficiência propostos neste trabalho, cercando empiricamente o problema de otimização da blocagem neste contexto. Outro aspecto a ser observado quanto à escolha de estratégias de agrupamento para sistemas de RE homomórficos é a quantidade de operações criptográficas são realizadas, tanto na codificação e decodificação quanto nas operações homomórficas. A introdução de uma métrica especializada em analisar a redução do número desse tipo de operação poderia refletir com mais exatidão os ganhos computacionais de cada abordagem em ambientes como o proposto pelo trabalho.

# Referências

- [1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys*, 51(4):1–35, July 2019.
- [2] Anmar A. Al-Janabi, Sufyan T. Faraj Al-Janabi, and Belal Al-Khateeb. Health-care privacy-preserving federated transfer learning using ckks-based homomorphic encryption and pyhfel tool. *Iraqi Journal For Computer Science and Mathematics*, 5(3):473–488, Aug. 2024.
- [3] Jeanne Cilliers Auke Rijpma and Johan Fourie. Record linkage in the cape of good hope panel\*. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 53(2):112–129, 2020.
- [4] Hendrik Ballhausen and Ludwig Christian Hinske. Federated secure computing. *Informatics*, 10(4):83, October 2023.
- [5] David Guy Brizan and Abdullah Uz Tansel. A. survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, 6(3), Jan 2006.
- [6] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 53(6), dec 2020.
- [7] CSIRO’s Data61. Python paillier library. <https://github.com/data61/python-paillier>, 2013.
- [8] Javier Diaz-Montes, Yu Xie, Ivan Roderio, Jaroslaw Zola, Baskar Ganapathysubramanian, and Manish Parashar. Federated computing for the masses—aggregating resources to tackle large-scale engineering problems. *Computing in Science Engineering*, 16(4):62–72, July 2014.
- [9] Enzo Fenoglio, Hirsh Pithadia, and Philip Treleaven. Federated computing. *SSRN Electronic Journal*, 2023.
- [10] Anja Gruenheid, Xin Luna Dong, and Divesh Srivastava. Incremental record linkage. *Proceedings of the VLDB Endowment*, 7(9):697–708, May 2014.

- [11] Shumin Han, Zikang Wang, Dengrong Shen, and Chuang Wang. A parallel multi-party privacy-preserving record linkage method based on a consortium blockchain. *Mathematics*, 12(12), 2024.
- [12] Shahidul Islam Khan, Abir Bin Ayub Khan, and Abu Sayed Md Latiful Hoque. Privacy preserved incremental record linkage. *Journal of Big Data*, 9(1):105, Nov 2022.
- [13] Hye-Chung Kum, Ashok Krishnamurthy, Ashwin Machanavajjhala, Michael K Reiter, and Stanley Ahalt. Privacy preserving interactive record linkage (PPIRL). *Journal of the American Medical Informatics Association*, 21(2):212–220, 11 2013.
- [14] Hye-Chung Kum, Eric D. Ragan, Gurudev Ilangovan, Mahin Ramezani, Qinbo Li, and Cason Schmit. Enhancing privacy through an interactive on-demand incremental information disclosure interface: Applying privacy-by-design to record linkage. In *Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security, SOUPS’19*, page 175–189, USA, 2019. USENIX Association.
- [15] Rajesh Kumar, WenYong Wang, Cheng Yuan, Jay Kumar, Zakria, He Qing, Ting Yang, and Abdullah Khan. Blockchain based privacy-preserved federated learning for medical images: A case study of covid-19 ct scans, 2021.
- [16] Junyi Li and Heng Huang. Faster secure data mining via distributed homomorphic encryption. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD ’20*, page 2706–2714, New York, NY, USA, 2020. Association for Computing Machinery.
- [17] Henrique Louzada. Incremental entity resolution through privacy-concerned federated architecture - source code. <https://github.com/henriquelouz/ufmg-ppgcc-paillier-entity-resolution>, 2024.
- [18] A. Maratea, A. Ciaramella, and G. P. Cianci. Record linkage of banks and municipalities through multiple criteria and neural networks. *PeerJ Comput Sci*, 6:e258, 2020.
- [19] Veena S. Kirubasri G. Thusnavis Bella Mary I. Udayakumar R. Mohandas, R. Federated learning with homomorphic encryption for ensuring privacy in medical data. *Indian Journal of Information Sources and Services*, 14:17–23, Jun. 2024.
- [20] Sarah Nechuta, Sutapa Mukhopadhyay, Shanthi Krishnaswami, Molly Golladay, and Melissa McPheeters. Record linkage approaches using prescription drug monitoring program and mortality data for public health analyses and epidemiologic studies. *Epidemiology*, 31(1), 2020.

- [21] Monique Ogburn, Claude Turner, and Pushkar Dahal. Homomorphic encryption. *Procedia Computer Science*, 20:502–509, 2013.
- [22] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. A survey of blocking and filtering techniques for entity resolution, 2020.
- [23] Luis Bernardo Pulido-Gaytan, Andrei Tchernykh, Jorge M. Cortés-Mendoza, Mikhail Babenko, and Gleb Radchenko. A survey on privacy-preserving machine learning with fully homomorphic encryption. In Sergio Nesmachnow, Harold Castro, and Andrei Tchernykh, editors, *High Performance Computing*, pages 115–129, Cham, 2021. Springer International Publishing.
- [24] S Rajashree, B Vineetha, Ami B Mehta, and Prasad B Honnavalli. Homomorphic encryption approach for string concatenation. In *2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, pages 267–272, 2022.
- [25] Shantanu Rane and Wei Sun. Privacy preserving string comparisons based on levenshtein distance. In *2010 IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2010.
- [26] James Scheibner, Marcello Ienca, and Effy Vayena. Health data privacy through homomorphic encryption and distributed ledger computing: an ethical-legal qualitative expert assessment study. *BMC Medical Ethics*, 23(1):121, Dec 2022.
- [27] Ahmed Soliman and Sanguthevar Rajasekaran. Firla: a fast incremental record linkage algorithm. *Journal of Biomedical Informatics*, 130:104094, 2022.
- [28] Tanyaporn Sridokmai and Somchai Prakancharoen. The homomorphic other property of paillier cryptosystem. In *2015 International Conference on Science and Technology (TICST)*, pages 356–359, 2015.
- [29] Kha Vo, Jitendra Jonnagaddala, and Siaw-Teng Liaw. Statistical supervised meta-ensemble algorithm for medical record linkage. *Journal of Biomedical Informatics*, 95:103220, 2019.
- [30] Ruidi Wei and Florian Kerschbaum. Cryptographically secure private record linkage using locality-sensitive hashing. *Proc. VLDB Endow.*, 17(2):79–91, October 2023.
- [31] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2):513–535, February 2023.
- [32] Steven Euijong Whang and Hector Garcia-Molina. Incremental entity resolution on rules and data. *The VLDB Journal*, 23(1):77–102, Feb 2014.

- 
- [33] Yunfan Ye, Shen Li, Fang Liu, Yonghao Tang, and Wanting Hu. Edgedfed: Optimized federated learning based on edge computing. *IEEE Access*, 8:209191–209198, 2020.

## Apêndice A

# Sistema Criptográfico de Paillier - Implementação

A Classe *Paillier* nos Algoritmos A.1, A.2 e A.3 deste apêndice propõem uma implementação em linguagem de alto nível do Sistema Criptográfico de Paillier.

Pela definição do sistema de Paillier,  $\lambda$  deve ser igual ao Mínimo Múltiplo Comum entre  $p - 1$  e  $q - 1$ . Aqui, simplificamos essa operação multiplicando estes termos (ou, calculando  $\phi$ ). Por se tratarem de números primos entre si, o seu MMC é obtido simplesmente através de sua multiplicação, já que não existem fatores comuns além de 1 entre eles. No Algoritmo A.1,  $\lambda$  é atribuído à variável *lmbda* e  $\phi$  à variável *phi*. Outra possível otimização presente no Algoritmo A.1 é a atribuição de  $g$ . Pela definição do sistema de Paillier,  $g$  deve ser um número aleatório tal que  $g \in Z_{n^2}^*$ . Como  $g$  não demanda validação, atribui-se o valor de  $n + 1$ , que atende à definição e não compromete a aleatoriedade da criptografia, visto que  $g$  é multiplicado por outra variável aleatória  $r$ .

---

```

1  Classe Paillier:
2      Funcao Paillier(): # Construtor
3          Paillier.pk, Paillier.sk = Paillier.gerar_chaves()
4
5      Funcao gerar_chaves():
6          p, q = gera numeros primos grandes
7          n = p * q
8          phi = (p-1) * (q-1)
9          lmbda = phi # ou, o MMC dos termos da variavel phi
10         g = n + 1 # ou, um numero inteiro aleatorio entre 0 e n^2
11         mu = phi^-1 modulo n
12
13         pk = (n, g)
14         sk = (lmbda, mu)
15         Retornar pk, sk
16     ...

```

---

Algoritmo A.1: Pseudocódigo com a implementação da classe Paillier. A função *gerar\_chaves* cria duas chaves, uma pública *pk* composta por  $n$  e  $g$ , e outra privada composta por  $\lambda$  e  $\mu$ , conforme definição do algoritmo.

---

```

1  Classe Paillier:
2      ...
3  Funcao encriptar(texto):
4      n, g = Paillier.pk
5      r = numero aleatorio entre 0 e n, primo a n
6      cifra = (g^texto modulo n^2) * (r^n modulo n^2) % (n^2)
7      Retornar cifra
8
9  Funcao descriptar(cifra):
10     n, _ = Paillier.pk
11     lmbda, mu = Paillier.sk
12     L = ((cifra^lmbda modulo n*n) - 1) // n # // parte inteira do
        quociente
13     texto = (L * mu) % n
14     Retornar texto
15     ...

```

---

Algoritmo A.2: Continuação do pseudocódigo com a implementação da classe Paillier. A função *encriptar* consulta os valores contidos na chave pública, gera um número aleatório  $r$  e calcula a cifra do texto original informado por parâmetro através da fórmula dada por definição. A função *descriptar* faz a operação reversa, consultando os valores das chaves pública e privada, e recuperando o texto original a partir da cifra informada por parâmetro ao implementar a fórmula dada por definição.

O problema que de fato eleva a complexidade computacional da geração de chaves no sistema de Paillier (Algoritmo A.1) é a atribuição de  $p$  e  $q$ . É necessário que estes números sejam suficientemente grandes para garantir a segurança do sistema. Ataques como o de força bruta, de medição física e utilizando o algoritmo de Fermat têm a sua taxa de sucesso inversamente interligada ao tamanho dos números que compõem a chave pública.

Porém, gerar dois números aleatórios suficientemente grandes e primos pode apresentar complicadores. Pela definição do sistema de Paillier, estes dois números devem ser gerados aleatoriamente e independentemente, assegurando inclusive que  $p * q$  e  $(p - 1) * (q - 1)$  sejam primos entre si. A geração de números aleatórios e a verificação de números primos entre si são tarefas de baixa complexidade, porém quanto maior os números gerados, mais complicada fica a sua prova de que são individualmente primos. Um algoritmo trivial de verificação da primalidade de  $n$ , por exemplo, itera por cada um dos números menores que  $\sqrt{n}$  e verifica o resto da divisão de  $n$  por eles, naturalmente de complexidade  $O(\sqrt{n})$ . Testes de primalidade mais sofisticados como o teste probabilístico de Miller-Rabin são mais eficientes nesse caso, pois dispõem de uma variável  $k$  de iterações, usada para calibrar a confiança da testagem. A complexidade do algoritmo de Miller-Rabin para verificação da primalidade de  $n$  é de  $O(k * \log^3 n)$ , o que assintoticamente é mais complexo que o algoritmo trivial, mas pode ser ajustado através de  $k$  para reverter a diferença e continuar satisfatoriamente preciso para números grandes.

---

```

1  Classe Paillier:
2      ...
3  Funcao inverso_modular(a, m):
4      m0 = m
5      x0 = 0
6      x1 = 1
7      Enquanto a > 1:
8          q = a // m
9          m = a % m
10         a = m
11         x0 = x1 - q * x0
12         x1 = x0
13     Se x1 < 0:
14         Retornar x1 + m0
15     Senao:
16         Retornar x1
17
18  Funcao subtracao_homomorfica(cifra1, cifra2):
19      n, _ = self.pk
20      cifra2_inv = Paillier.inverso_modular(cifra2, n^2)
21      Retornar (cifra1 * cifra2_inv) % n^2

```

---

Algoritmo A.3: Continuação do pseudocódigo com a implementação da classe Paillier. A função *subtracao\_homomorfica* realiza a multiplicação modular entre uma primeira cifra informada por parâmetro e o inverso modular de uma segunda cifra, também informada por parâmetro.

No Algoritmo A.2, observamos a implementação dos métodos de criptografia. Aqui, notamos a representação das permissões de acesso às chaves. O método *encriptar* acessa os valores de  $n$  e  $g$  da chave pública do sistema, o que representa que, em uma implementação distribuída, este método estaria disponível publicamente. O método *desencriptar* por sua vez acessa tanto o valor  $n$  da chave pública quanto os valores de  $\lambda$  e  $\mu$  da chave privada. Em uma implementação distribuída, isso representa que este método só estaria disponível para quem detivesse a chave privada do sistema. A complexidade mais acentuada destes procedimentos é a geração do número aleatório  $r \in N_n^*$  /  $r$  é primo a  $n$ . Neste caso, é utilizada a função Totiente de Euler  $\phi(n)$  para obtenção de números coprimos e menores que  $n$ , apresentando uma complexidade de  $O(\log n)$ .

O Algoritmo A.3 propõe uma implementação para a subtração homomórfica através do método *subtracao\_homomorfica*. Aqui, o cálculo da subtração se inicia através da leitura do valor de  $n$  da chave pública e o cálculo da multiplicação entre a primeira cifra e o inverso modular da segunda cifra módulo  $n^2$ . A complexidade dessa operação se encontra no cálculo do inverso modular entre dois números, aqui implementado no método *inverso\_modular*. Ele utiliza da definição do algoritmo estendido de Euclides para calcular o inverso modular de  $a \bmod m$  informados por parâmetro em tempo  $O(\log(m))$ .

## Apêndice B

# Sistema Criptográfico de Paillier - Corretude

O homomorfismo do Sistema Criptográfico de Paillier é um recurso que permite o uso de operações matemáticas em dados cifrados, o que é de utilidade em vários tipos de sistemas computacionais. Sistemas de processamento distribuído, operações financeiras e aprendizado de máquina federado são algumas das diversas aplicações nas quais esse tipo de sistema criptográfico desempenha um papel importante em termos de cibersegurança e confidencialidade de informações. Podemos verificar a corretude deste sistema em cada um de seus métodos, estes que se baseiam nas propriedades dos números primos e em operações modulares para garantir a propriedade homomórfica do sistema.

### B.1 Segurança da Criptografia e Garantia de Homomorfismo

A operação de criptografia é realizada através da fórmula:

$$c = g^m \times r^n \bmod n^2,$$

em que  $g$  e  $r$  são variáveis aleatórias, sendo  $g$  parte da chave pública de criptografia. Neste caso, a adição da variável aleatória  $r$  garante que, independente de  $g$ , ainda haja distinção entre cifras criadas com chaves diferentes para textos iguais. Essa premissa da criptografia é chamada de Função de Mão Única, uma técnica que através da adição de aleatoriedade assegura que a cifra só consiga ser revertida para seu valor original através de uma chave privada.

Além de garantir a segurança dos dados cifrados, outro dever de um sistema criptográfico homomórfico é garantir o seu homomorfismo, i.e., a sua capacidade de ser operado mesmo depois de criptografado. Nesse sentido, a variável  $n$  desempenha o papel de manter o homomorfismo do sistema, uma vez que o produto da multiplicação entre duas cifras  $c_1$

e  $c_2$  (o que, no sistema de Paillier, significa a adição entre os textos originais) pode ser interpretado como:

$$c_1 \times c_2 = (g^{m_1} \times r_1^n) \times (g^{m_2} \times r_2^n) \bmod n^2 = g^{m_1+m_2} \times (r_1 \times r_2)^n \bmod n^2.$$

Como observado, a aleatoriedade de  $r_1 \times r_2$  é mantida em  $(r_1 \times r_2)^n$ , e portanto o resultado de  $m_1 + m_2$  não é comprometido. Da mesma maneira, quando se trata da subtração entre dois valores criptografados em  $c_1$  e  $c_2$  descrita como a multiplicação entre a primeira cifra pelo inverso modular da segunda  $c_1 \times (c_2^{-1}) \bmod n^2$ , é preciso encontrar um valor para  $c^{-1}$  tal que  $c \times c^{-1} \equiv 1 \bmod n^2$ . Abstraindo a complexidade dessa operação, temos que a inversa modular é garantida pelo grupo multiplicativo  $Z_{n^2}^*$ , o que indica que a multiplicação  $c_1 \times (c_2^{-1}) \bmod n^2$  ocorre dentro deste mesmo grupo  $Z_{n^2}^*$ . Isso garante que o resultado também se encontra neste grupo e, portanto,  $m_1 - m_2$  não é comprometido.

## B.2 Corretude da Descriptografia

A corretude da operação inversa à criptografia é garantida através do Teorema de Fermat-Euler, que afirma que se  $n \in N$ , e  $a \in N / a$  é primo a  $n$ , então:

$$a^{\phi(n)} \equiv 1 \bmod n.$$

Dessa forma, temos que o texto original  $m$  é obtido através de:

$$m = L(c^\lambda \bmod n^2) \times \mu \bmod n.$$

Para realizar essa operação, temos que descobrir o valor de  $c^\lambda$ . Partindo da fórmula da operação de criptografia, temos que:

$$c^\lambda = ((1+n)^m \times r^n)^\lambda \bmod n^2,$$

ou:

$$c^\lambda = (1+n)^{m\lambda} \times r^{n\lambda} \bmod n^2.$$

Considerando que  $\lambda$  é igual à Função Totiente de  $n$  (ou  $\lambda = \phi(n)$ ), podemos simplificar a expressão de  $c^\lambda$  através do Teorema de Fermat-Euler.

Considerando a propriedade da Função Totiente de Euler:

$$\phi(n^2) = n \times \phi(n),$$

e a fórmula do Teorema de Fermat-Euler, obtemos:

$$r^{n\phi(n)} \bmod n^2 = r^{\phi(n^2)} \bmod n^2 = 1.$$

$c^\lambda$  neste caso pode ser simplificado como:

$$c^\lambda = (1 + n)^{m\lambda} \bmod n^2.$$

Através do Binômio de Newton, observamos que essa expressão pode ser escrita como:

$$c^\lambda = 1 + nm\lambda \bmod n^2,$$

uma vez que as potências iguais ou superiores a  $n^2$  serão sempre divisíveis por  $n^2$ . Aplicando a função  $L(u) = (u - 1)/n$  ao  $c^\lambda \bmod n^2$  obtido acima, temos que:

$$L(c^\lambda \bmod n^2) = (1 + nm\lambda - 1)/n = m\lambda.$$

Retornando à função original da descriptografia, multiplicamos  $m\lambda$  por  $\mu$ :

$$L(c^\lambda \bmod n^2) \times \mu = m\lambda \times \mu.$$

Considerando que  $\mu = \phi(n)^{-1} \bmod n$ , i.e.,  $\mu$  é o inverso da Função Totiente em  $\lambda$  no módulo de  $n$ , verificamos finalmente que:

$$L(c^\lambda \bmod n^2) \times \mu = m \times \phi(n) \times \phi(n)^{-1} \bmod n = m.$$