

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA E INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM ENGENHARIA COMPUTACIONAL

Andrew Buzinari d Ornellas Vieira

**Avaliação de Privacidade e Eficiência em Sistemas de Recomendação
Federados com Criptografia Homomórfica**

Juiz de Fora

2026

Buzinari d Ornellas Vieira, Andrew.

Avaliação de Privacidade e Eficiência em Sistemas de Recomendação Federados com Criptografia Homomórfica / Andrew Buzinari d Ornellas Vieira. - 2026.

40 f.

Orientador: Alex Borges Vieira Orientador

Trabalho de Conclusão de Curso (Bacharelado) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia e Instituto de Ciências Exatas. Bacharelado em Engenharia Computacional, 2026.

1. Aprendizado Federado. 2. Criptografia Homomórfica. 3. Sistemas de Recomendação. 4. Privacidade por Design. I. Orientador, Alex Borges Vieira, orient. II. Título.

Andrew Buzinari d Ornellas Vieira

**Avaliação de Privacidade e Eficiência em Sistemas de Recomendação
Federados com Criptografia Homomórfica**

Trabalho de Conclusão de Curso apresentado
ao corpo docente do curso de graduação em
Engenharia Computacional da Faculdade de
Engenharia e Instituto de Ciências Exatas
da Universidade Federal de Juiz de Fora
como requisito parcial à obtenção do grau de
Bacharel em Engenharia Computacional.

Orientador: Prof. D.Sc. Alex Borges Vieira Orientador

Juiz de Fora

2026

Andrew Buzinari d Ornellas Vieira

**Avaliação de Privacidade e Eficiência em Sistemas de Recomendação
Federados com Criptografia Homomórfica**

Trabalho de Conclusão de Curso apresentado
ao corpo docente do curso de graduação em
Engenharia Computacional da Faculdade de
Engenharia e Instituto de Ciências Exatas
da Universidade Federal de Juiz de Fora
como requisito parcial à obtenção do grau de
Bacharel em Engenharia Computacional.

Aprovado em 13 de julho de 2026.

BANCA EXAMINADORA

Prof. D.Sc. Alex Borges Vieira Orientador - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Edelberto Franco Silva
Universidade Federal de Juiz de Fora

Prof. D.Sc. Luciano Jerez Chaves
Universidade Federal de Juiz de Fora

*Ao meu pai e à memória de
minha tia-avó, exemplos de amor,
dignidade e perseverança.*

AGRADECIMENTOS

Agradeço, primeiramente, ao meu pai, Deusdedit Paiva Vieira. Um grande homem, minha maior inspiração e a quem atribuo tudo o que me tornei. Agradeço por ter me ensinado o valor da honra: a coragem de fazer o que é certo, ainda que não seja conveniente, e por ter me guiado em cada passo com os ensinamentos que hoje definem meu caráter.

À memória de minha tia-avó, que desempenhou com dedicação o papel de mãe. Pelo seu exemplo silencioso, ela me concedeu as maiores lições de vida: ensinou-me que o trabalho dignifica e que o amor não se prova apenas com afeto, mas pela renúncia do próprio conforto em prol do cuidado ao próximo. Mesmo com pouco estudo formal, sua sabedoria foi imensa e sua presença será eterna em meus pensamentos.

Aos amigos que a universidade me presenteou, Bruno Moraes, Jhuan Olimpio e Felipe Iglesias. Pela parceria constante na execução dos trabalhos acadêmicos e pelo apoio mútuo nos momentos de descontração e conforto pessoal, tornando esta jornada acadêmica muito mais leve e amistosa.

Por fim, expresso minha profunda gratidão ao meu orientador e professor, Alex Borges Vieira. Além da excelência na orientação técnica e das ideias fundamentais que aprimoraram este trabalho, agradeço pela sua integridade e por ter sido um confidente essencial nos momentos de maior dificuldade.

*“O sucesso é ir de fracasso
em fracasso sem perder o
entusiasmo.”*

(Winston Churchill)

RESUMO

Sistemas de recomendação dependem tradicionalmente da centralização de dados sensíveis, gerando conflitos com legislações modernas de proteção de dados, como a LGPD. O Aprendizado Federado (FL) mitiga este risco ao descentralizar o treino, mas os gradientes partilhados continuam vulneráveis a ataques de inferência. Este trabalho investiga a integração da Criptografia Homomórfica (esquema CKKS) num modelo Federado de Fatoração de Matrizes para garantir privacidade estrita. Utilizando o framework Flower e a biblioteca TenSEAL, avaliamos o algoritmo FedProx para lidar com a heterogeneidade dos dados (Non-IID) inerente aos sistemas de recomendação. Os resultados experimentais demonstram que a precisão preditiva é totalmente preservada sob criptografia, com o ruído de aproximação homomórfica a atuar como um regularizador implícito que pode estabilizar a convergência. Contudo, a camada criptográfica impõe um elevado custo computacional e expansão de dados, expondo trade-offs críticos para implantações escaláveis em tempo real.

Palavras-chave: Aprendizado Federado. Criptografia Homomórfica. Sistemas de Recomendação. Privacidade por Design.

ABSTRACT

Recommender systems traditionally rely on the centralization of sensitive data, generating conflicts with modern data protection regulations, such as the LGPD. Federated Learning (FL) mitigates this risk by decentralizing the training process, but shared gradients remain vulnerable to inference attacks. This work investigates the integration of Homomorphic Encryption (CKKS scheme) into a Federated Matrix Factorization model to ensure strict privacy. Using the Flower framework and the TenSEAL library, we evaluated the FedProx algorithm to handle the data heterogeneity (Non-IID) inherent to recommender systems. Experimental results demonstrate that predictive accuracy is fully preserved under encryption, with the homomorphic approximation noise acting as an implicit regularizer that can stabilize convergence. However, the cryptographic layer imposes a high computational cost and data expansion, exposing critical trade-offs for real-time scalable deployments.

Keywords: Federated Learning. Homomorphic Encryption. Recommender Systems. Privacy by Design.

LISTA DE ILUSTRAÇÕES

2.1	Esquema de fatoração de matrizes em sistemas de recomendação	16
2.2	Arquitetura Cliente-Servidor no Aprendizado Federado	17
3.1	Diagrama da arquitetura proposta	22

LISTA DE TABELAS

3.1	Hiperparâmetros do Algoritmo de Recomendação	25
3.2	Parâmetros do Esquema CKKS	29
3.3	Configurações experimentais avaliadas	32
4.1	Resultados Consolidados das Métricas de Avaliação	33
4.2	Tempos Totais de Treinamento (50 Rodadas Federadas)	36

LISTA DE ABREVIACÕES E SIGLAS

LGPD	Lei Geral de Proteção de Dados
FL	<i>Federated Learning</i> (Aprendizado Federado)
CKKS	Cheon–Kim–Kim–Song
FedAvg	<i>Federated Averaging</i>
FedProx	<i>Federated Proximal</i>
F1-score	Medida harmônica entre Precisão e Recall
Recall	Taxa de Verdadeiros Positivos
MF	Matrix Factorization (Fatoração de Matrizes)
TT	Train-Test Split (Treino-Teste Divisão)
LR	Learning Rate (taxa de aprendizado)
SGD	Stochastic Gradient Descent (Descida de Gradiente Estocástico)
HE	Homomorphic Encryption (Criptografia Homomórfica)
Non-IID	Not Independent and Identically Distributed

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Motivação e Contextualização	13
1.2	O Problema	13
1.3	Contribuições do Trabalho	13
1.4	Estrutura	14
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	Sistemas de Recomendação e Fatoração de Matrizes	15
2.2	Aprendizado Federado	17
2.2.1	Heterogeneidade de Dados e o Algoritmo FedProx	18
2.3	Criptografia Homomórfica e Esquema CKKS	19
3	METODOLOGIA	21
3.1	Arquitetura Proposta	21
3.2	Ambiente utilizado para experimentação	22
3.3	Modelagem da Interação e Critérios de Relevância	23
3.4	Base de Dados e Pré-processamento	23
3.5	Algoritmo de Recomendação e Hiperparâmetros	24
3.6	Orquestração Federada com o Framework Flower	26
3.7	Implementação da Estratégia FedProx	27
3.8	Camada de Criptografia Homomórfica (TenSEAL)	28
3.9	Cenários de Avaliação e Métricas	31
4	RESULTADOS E DISCUSSÕES	33
4.1	Validação do Modelo	33
4.2	O Impacto da Heterogeneidade e a Eficácia do FedProx	34
4.3	Orquestração Segura e o Ruído como Regularizador Implícito	35
4.4	Avaliação do Tempo de Execução	36

5	CONCLUSÕES E TRABALHOS FUTUROS.....	38
5.1	Conclusões Gerais	38
5.2	Limitações e Trabalhos Futuros	38

1 INTRODUÇÃO

1.1 MOTIVAÇÃO E CONTEXTUALIZAÇÃO

Sistemas de recomendação desempenham um papel central na personalização de conteúdo em plataformas digitais. Tradicionalmente, a eficácia desses algoritmos, como a Fatoração de Matrizes, depende da centralização de dados comportamentais, o que gera conflitos diretos com legislações de proteção de dados (*Lei Geral de Proteção de Dados* (LGPD)) e riscos de privacidade. O desafio atual da área consiste em viabilizar o treinamento colaborativo desses modelos sem a necessidade de expor ou centralizar os dados sensíveis dos usuários.

1.2 O PROBLEMA

O Aprendizado Federado (*Federated Learning*) mitiga esse risco ao processar os dados localmente nos dispositivos. Contudo, a simples descentralização não garante anonimato total, pois a inspeção dos gradientes compartilhados pode revelar informações sobre os dados de treino. A solução para blindar esse processo é a Criptografia Homomórfica, que permite a agregação matemática de modelos cifrados. O problema prático abordado neste trabalho é a complexidade de engenharia envolvida na integração destas tecnologias. Investigamos a viabilidade de converter uma simulação teórica para um framework federado consolidado, analisando o impacto na precisão das recomendações ao introduzir criptografia robusta (*Cheon–Kim–Kim–Song* (CKKS)) e diferentes estratégias de agregação.

1.3 CONTRIBUIÇÕES DO TRABALHO

Neste trabalho, desenvolvemos e avaliamos uma implementação de um Sistema de Recomendação Federado Seguro. As principais contribuições técnicas podem ser resumidas da seguinte forma:

- **Arquitetura e Implementação:** Sistematizamos a migração de uma simulação manual para o framework Flower, validando a integração entre algoritmos de

recomendação e ferramentas de aprendizado federado utilizadas na indústria. Esse processo envolveu a superação de desafios relacionados à serialização e ao gerenciamento de estruturas de dados complexas no contexto federado.

- **Comparação de Estratégias (FedAvg vs. FedProx):** Implementamos e comparamos o algoritmo clássico FedAvg (*Federated Averaging*) com o FedProx (*Federated Proximal*), analisando especificamente o impacto do termo proximal (μ) sobre a qualidade final do modelo. A avaliação foi conduzida por meio de métricas como *F1-score*, *Recall* e Precisão, permitindo uma comparação direta com a abordagem tradicional.
- **Privacidade com TenSEAL:** Integramos a biblioteca TenSEAL ao fluxo de treinamento federado utilizando o esquema criptográfico CKKS. Avaliamos o *trade-off* entre privacidade e utilidade, investigando se as aproximações numéricas inerentes à criptografia homomórfica impactam negativamente a precisão do sistema de recomendação quando comparado ao treinamento em texto plano.

1.4 ESTRUTURA

O trabalho organiza-se da seguinte forma: O Capítulo 2 fornece a fundamentação teórica sobre Fatoração de Matrizes, Aprendizado Federado e o esquema criptográfico CKKS. O Capítulo 3 detalha a metodologia, descrevendo a configuração dos experimentos no framework Flower e os parâmetros utilizados. O Capítulo 4 discute os resultados, comparando as métricas de precisão e desempenho do modelo entre os diferentes cenários propostos. Por fim, O Capítulo 5 apresenta as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais que baseiam a arquitetura desenvolvida neste trabalho. Primeiramente, são abordados os Sistemas de Recomendação com foco na técnica de fatoração de matrizes. Em seguida, discute-se o paradigma do aprendizado federado e suas estratégias de otimização em cenários heterogêneos. Por fim, apresenta-se a criptografia homomórfica, com ênfase no esquema CKKS, essencial para garantir a confidencialidade das operações.

2.1 SISTEMAS DE RECOMENDAÇÃO E FATORAÇÃO DE MATRIZES

Sistemas de recomendação são ferramentas de filtragem que buscam prever a preferência que um usuário atribuiria a um determinado item Motta *et al.* (2011). Dentre as abordagens mais famosas, destaca-se a filtragem colaborativa baseada em fatoração de matrizes (*Matrix Factorization* - MF) Zhang (2022), que mapeia usuários e itens em um espaço conjunto de dimensão reduzida (fatores latentes).

A premissa da fatoração de matrizes é decompor uma matriz de avaliações esparsa R , de dimensões $U \times I$ (onde U é o número de usuários e I o número de itens), em duas matrizes menores, sendo uma matriz de características dos usuários P (de dimensão $U \times F$) e uma matriz de características dos itens Q (de dimensão $I \times F$), onde F representa o número de fatores latentes.

A predição da nota $\hat{r}_{ui}$ que um usuário u daria a um item i é obtida pelo produto escalar de seus respectivos vetores de fatores latentes:

$$\hat{r}_{ui} = p_u \cdot q_i^T \quad (2.1)$$

A Figura 2.1 ilustra esse processo, mostrando como a matriz de avaliações R é decomposta nas matrizes P e Q , e como a multiplicação entre essas matrizes permite reconstruir uma aproximação $\hat{R}$, incluindo a predição das notas desconhecidas.

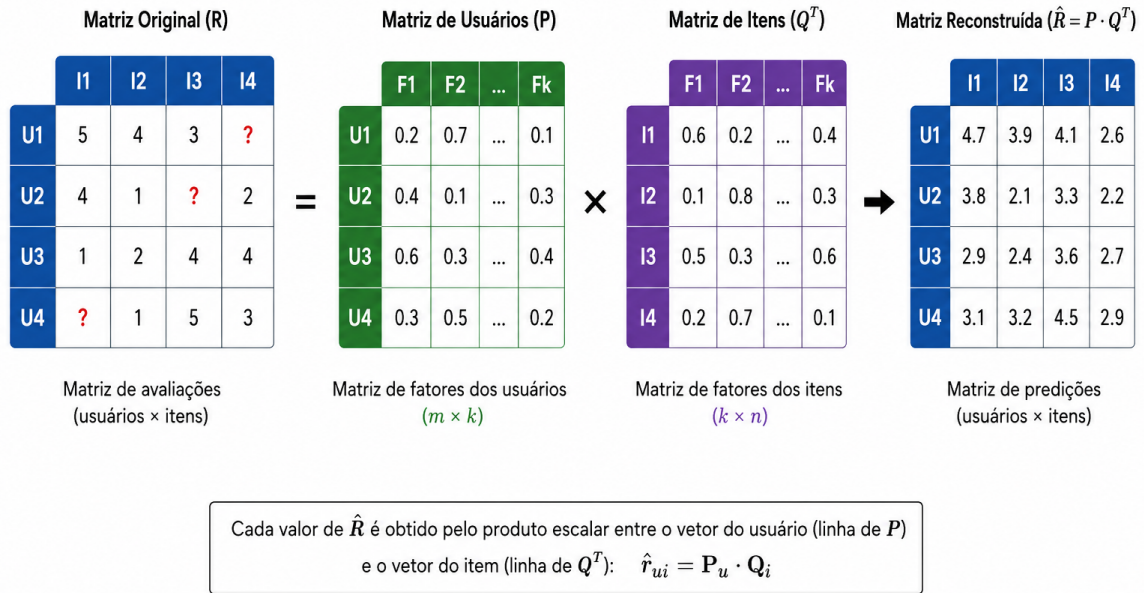


Figura 2.1: Esquema de fatoração de matrizes em sistemas de recomendação

Para aprender essas matrizes, o modelo minimiza o erro entre as notas reais conhecidas r_{ui} e as notas preditas $\hat{r}_{ui}$. A otimização é rotineiramente feita utilizando o algoritmo de Descida de Gradiente Estocástico (*Stochastic Gradient Descent* - (SGD)). Para evitar o sobreajuste (*overfitting*), introduz-se um termo de regularização λ . A função objetivo a ser minimizada é dada por: Onde K é o conjunto de pares (u, i) para os quais as notas são conhecidas.

$$\min_{P, Q} \sum_{(u, i) \in K} (r_{ui} - p_u \cdot q_i^T)^2 + \lambda(||p_u||^2 + ||q_i||^2) \quad (2.2)$$

As regras de atualização para os vetores em cada iteração, controladas por uma taxa de aprendizado α (*learning rate*), são: Onde $e_{ui} = r_{ui} - \hat{r}_{ui}$ é o erro de predição.

$$p_u \leftarrow p_u + \alpha(e_{ui} \cdot q_i - \lambda \cdot p_u) \quad (2.3)$$

$$q_i \leftarrow q_i + \alpha(e_{ui} \cdot p_u - \lambda \cdot q_i) \quad (2.4)$$

No contexto deste trabalho, os vetores p_u contêm as preferências dos usuários e devem permanecer estritamente em seus dispositivos locais, enquanto a matriz Q é o modelo global compartilhado e iterativamente atualizado.

2.2 APRENDIZADO FEDERADO

O Aprendizado Federado (*Federated Learning* - FL) é um paradigma de aprendizado de máquina descentralizado introduzido por McMahan *et al.* (2017). Diferente da abordagem tradicional, onde os dados são centralizados em um servidor, no FL os dados brutos permanecem nos dispositivos locais dos clientes. O treinamento do modelo ocorre localmente, e apenas as atualizações matemáticas (gradientes ou novos pesos) são transmitidas ao servidor central.

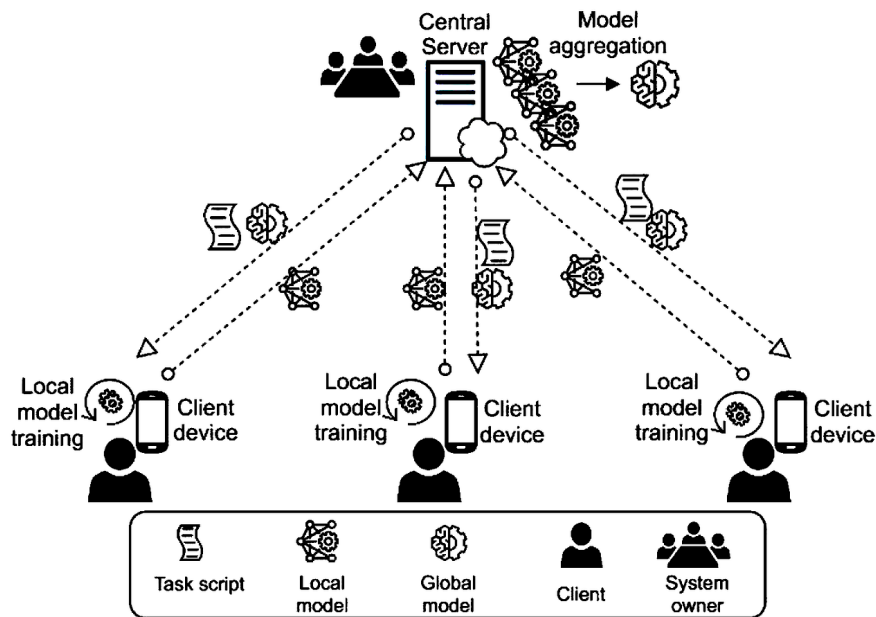


Figura 2.2: Arquitetura Cliente-Servidor no Aprendizado Federado

A Figura 2.2 apresenta a arquitetura cliente-servidor típica do Aprendizado Federado. Nesse paradigma, um servidor central é responsável por coordenar o treinamento distribuído do modelo global, enquanto os dispositivos clientes mantêm seus dados armazenados localmente. Inicialmente, o servidor envia o modelo global e as instruções de treinamento aos clientes participantes. Cada cliente realiza o treinamento utilizando exclusivamente seus próprios dados, produzindo um modelo local atualizado. Em seguida,

apenas os parâmetros ou atualizações do modelo são enviados de volta ao servidor, sem que os dados brutos sejam compartilhados. Por fim, o servidor executa um algoritmo de agregação combinando as contribuições recebidas para gerar uma nova versão do modelo global, que será redistribuída aos clientes nas próximas rodadas de treinamento.

A estratégia de agregação mais utilizada do FL é o algoritmo FedAvg, inserida por McMahan *et al.* (2017). O servidor seleciona uma fração de clientes a cada rodada t , envia o modelo global atual w_t , e cada cliente realiza múltiplas épocas de treinamento local via SGD. Posteriormente, o servidor calcula a média ponderada dos modelos recebidos para formar o novo modelo global w_{t+1} : Onde n_k é o número de amostras do cliente k e n é o total de amostras usadas na rodada.

$$w_{t+1} = \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k \quad (2.5)$$

2.2.1 Heterogeneidade de Dados e o Algoritmo FedProx

Apesar do sucesso do FedAvg, a literatura aponta que ele sofre severa degradação de desempenho em cenários onde os dados não são independentes e identicamente distribuídos (Non-IID), como relata Orlandi (2023). Em sistemas de recomendação, o cenário Non-IID é comum. Um usuário pode ter milhares de avaliações exclusivas de filmes de terror, enquanto outro possui apenas avaliações de filmes de comédia. Se esses usuários treinarem modelos localmente de forma isolada, as matrizes resultantes divergirão fortemente do ótimo global.

Para mitigar essa divergência local, Li *et al.* (2020) propuseram o algoritmo FedProx. O FedProx introduz um termo de regularização proximal à função objetivo local do cliente. Esse termo restringe o impacto das atualizações locais, forçando o modelo local a não se distanciar excessivamente do modelo global original (usado como âncora). A nova função objetivo local passa a ser: Onde $F_k(w)$ é a perda empírica padrão do cliente k , w^t é o modelo global recebido no início da rodada, e μ é o hiperparâmetro que controla a força do termo proximal. No contexto do cálculo do gradiente local, essa alteração exige adicionar a penalização $\mu(w - w^t)$ diretamente à regra de atualização.

$$\min_w h_k(w) = F_k(w) + \frac{\mu}{2} \|w - w^t\|^2 \quad (2.6)$$

2.3 CRIPTOGRAFIA HOMOMÓRFICA E ESQUEMA CKKS

Embora o aprendizado federado evite o compartilhamento de dados brutos, pesquisas demonstraram que a interceptação das atualizações do modelo (a matriz Q trafegada pela rede) permite ataques de inferência capazes de extrair informações sensíveis sobre o comportamento dos usuários, como é citado por Batista *et al.* (2024). Para blindar esse processo, torna-se interessante o uso da criptografia homomórfica (*Homomorphic Encryption* - HE).

A HE é uma forma especial de criptografia que permite realizar cálculos matemáticos diretamente sobre dados cifrados, sem a necessidade de uma chave de descryptografia durante o processamento. Se denotarmos a função de encriptação como $E(\cdot)$, a propriedade homomórfica garante que:

$$E(m_1) + E(m_2) = E(m_1 + m_2) \quad (2.7)$$

Isto significa que o servidor pode receber as matrizes Q criptografadas de vários clientes e somá-las. O resultado será um novo vetor criptografado que, quando descryptografado pelo detentor da chave privada, revelará a soma exata dos modelos originais. O servidor atua como um agregador “cego”, não tendo acesso a nenhum dado em texto plano.

A maioria dos esquemas clássicos de HE opera estritamente sobre números inteiros. No entanto, os pesos e gradientes extraídos de algoritmos de *machine learning* (como o SGD) são essencialmente compostos por números reais. O uso de esquemas puramente inteiros exigiria técnicas complexas e custosas.

Para resolver esta limitação, Cheon *et al.* (2017) introduziram o esquema **CKKS**. O CKKS suporta nativamente a aritmética de números reais aproximados. Ele opera multiplicando a mensagem original por um fator de escala (*scale* Δ) e, em seguida, arredondando o resultado para o inteiro mais próximo antes de encriptar. As operações homomórficas geram o crescimento indesejado da escala e a adição de um pequeno ruído

matemático para garantir a segurança da cifra, como aborda Vinko *et al.* (2026).

A aplicação do esquema CKKS impõe a parametrização rigorosa de variáveis criptográficas, notadamente o grau do módulo polinomial (N) e o tamanho dos bits de coeficiente (q). O grau N (tipicamente 4096 ou 8192) determina o tamanho do vetor empacotado e afeta diretamente a segurança e a latência computacional. Parâmetros maiores garantem maior segurança, mas geram *payloads* massivos.

3 METODOLOGIA

Este capítulo detalha a arquitetura do sistema desenvolvido, as ferramentas utilizadas e as configurações dos experimentos. A metodologia adota uma abordagem experimental quantitativa, dividida na preparação dos dados, configuração do algoritmo de recomendação, orquestração federada e integração da camada criptográfica. Por fim, são descritos os cenários de avaliação e as métricas utilizadas para validação do modelo.

3.1 ARQUITETURA PROPOSTA

A arquitetura proposta baseia-se em um sistema de Aprendizado Federado Seguro para recomendação, composto por duas entidades principais: o **Servidor Central** (agregador) e os **Clientes** (detentores dos dados). O fluxo de operação foi desenhado para garantir que a privacidade seja preservada desde a concepção (*Privacy by Design*).

O protocolo de comunicação segue a ordem cronológica abaixo:

1. **Setup Criptográfico:** O servidor gera o contexto da Criptografia Homomórfica (HE) e distribui a chave pública para os clientes.
2. **Broadcast do Modelo:** O servidor envia a matriz global de itens Q para os clientes selecionados.
3. **Treinamento Local:** O cliente realiza a fatoração local atualizando sua matriz de usuários P (que nunca sai do dispositivo) e gerando um novo gradiente para Q .
4. **Criptografia e Envio:** O cliente cifra o gradiente da matriz Q usando o esquema CKKS e envia o texto cifrado (*ciphertext*) ao servidor.
5. **Agregação Cega:** O servidor realiza a soma homomórfica das atualizações cifradas, calculando a nova média global sem nunca visualizar os dados em texto plano.

Podemos observar a arquitetura proposta de forma a elucidar melhor funcionamento do modelo na Figura 3.1

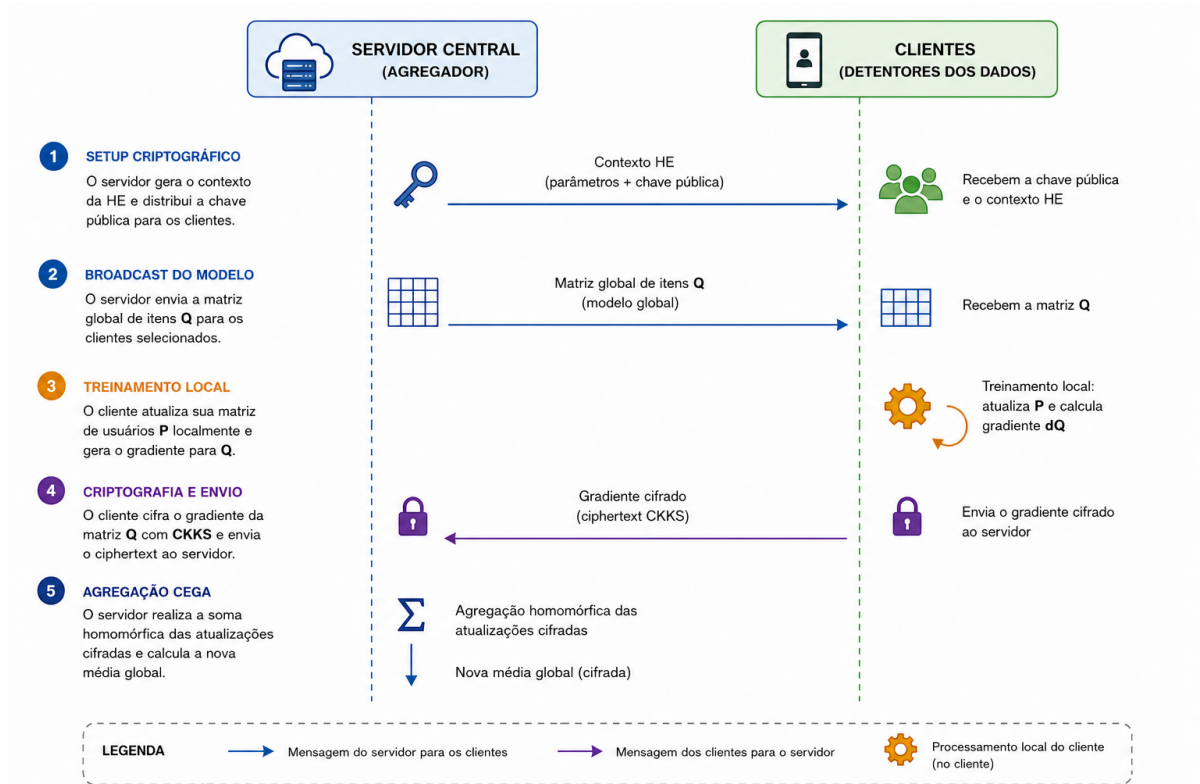


Figura 3.1: Diagrama da arquitetura proposta

3.2 AMBIENTE UTILIZADO PARA EXPERIMENTAÇÃO

Para a realização dos experimentos e coleta dos dados de desempenho, foi utilizada uma estação de trabalho padronizada ¹. As especificações técnicas do hardware e do ambiente de software utilizados durante as simulações estão detalhadas a seguir:

- **Processador:** Intel Core i5-10600KF, 4,1 GHz
- **Memória RAM:** 32 GB DDR4
- **Sistema Operacional:** Windows 10
- **Python:** versão 3.12.3
- **TenSEAL:** versão compatível com CKKS

¹O código-fonte completo da implementação deste trabalho, incluindo a orquestração federada e a camada criptográfica, está disponível em: https://github.com/Andrew-proj/F1_He_MovieLensSmall

3.3 MODELAGEM DA INTERAÇÃO E CRITÉRIOS DE RELEVÂNCIA

O sistema proposto atua sobre uma matriz de interações $R_{U \times I}$, onde cada entrada r_{ui} representa a preferência expressa de um usuário por um item. Para aumentar a precisão do modelo e focar no aprendizado de preferências positivas, define-se um limiar de relevância denominado *min_rating*.

O conjunto de treinamento é filtrado para considerar apenas interações onde:

$$\mathcal{K}_{train} = \{(u, i) \in R \mid r_{ui} \geq \text{min_rating}\} \quad (3.1)$$

A escolha do valor $\text{min_rating} = 3.0$ foi definida como uma decisão metodológica deste trabalho, considerando a escala de avaliação de 1 a 5 utilizada pelo conjunto de dados. Notas inferiores a 3.0 costumam indicar insatisfação do usuário, o que poderia introduzir ruído nos fatores latentes que buscam capturar o que o usuário efetivamente gostaram. Além disso, o usuário pode realizar múltiplas avaliações, mas apenas o histórico que atende a este critério compõe o vetor de entrada para a fatoração de matrizes.

3.4 BASE DE DADOS E PRÉ-PROCESSAMENTO

Para o treinamento e avaliação do sistema de recomendação, foi utilizado o dataset público **MovieLens ml-latest-small**, amplamente adotado na literatura para tarefas de filtragem colaborativa. O conjunto de dados contém avaliações de filmes feitas por usuários, contendo os identificadores do usuário, do filme e a nota atribuída.

Durante a etapa de avaliação, foram considerados apenas os usuários que possuíam ao menos seis avaliações relevantes no conjunto de teste, sendo cinco utilizadas como histórico de entrada e o restante como verdade fundamental (*ground truth*). Usuários com menos avaliações foram excluídos do cálculo das métricas, mas participaram normalmente do treinamento federado.

A base de dados foi dividida utilizando a técnica de *Holdout* (Train-Test Split - TT) por usuário, reservando 90% dos usuários para o treinamento federado e 10% exclusivamente para o teste do modelo global (variável: $TEST_SIZE = 0.1$).

3.5 ALGORITMO DE RECOMENDAÇÃO E HIPERPARÂMETROS

O treinamento local em cada cliente foi implementado utilizando a Fatoração de Matrizes otimizada por meio da Descida de Gradiente Estocástico em Mini-Lotes (*Mini-Batch SGD*). A adoção de mini-lotes, aliada a operações vetorizadas da biblioteca NumPy, acelerou a convergência local e otimizou o uso de memória.

Para garantir a convergência estável do modelo e evitar o sobreajuste aos dados locais de treinamento, o algoritmo foi aprimorado com duas técnicas fundamentais: a **Regularização L2** e o otimizador **Momentum**.

A Fatoração de Matrizes, por sua natureza esparsa, é altamente suscetível ao *overfitting*. Para mitigar este problema, aplicou-se a Regularização L2, que adiciona uma penalidade proporcional ao quadrado da magnitude dos pesos diretamente no cálculo do gradiente (Goodfellow *et al.* (2016)). O hiperparâmetro λ (`regularization = 0.01`) restringe o crescimento excessivo dos fatores latentes, forçando o modelo a aprender padrões genéricos em vez de memorizar os dados locais.

Adicionalmente, o SGD tradicional frequentemente sofre com oscilações bruscas em regiões onde a superfície de erro apresenta curvaturas acentuadas (vales). Para acelerar o treinamento nessas direções e amortecer as oscilações, implementou-se o otimizador *Momentum* (Ruder (2016)). O método atua acumulando uma média móvel exponencial dos gradientes passados, controlada pelo fator de decaimento $\beta = 0.9$. O Algoritmo 3.1 demonstra a integração destas duas técnicas no código desenvolvido.

A taxa de aprendizado (LR) sofreu um decaimento exponencial a cada época local para garantir que os passos de atualização se tornassem menores nas iterações finais, promovendo um ajuste fino nas matrizes de características.

A lei de decaimento aplicada, fundamentada nas técnicas de otimização para descida de gradiente discutidas por Ruder (2016), é dada por: onde α_0 é a taxa de aprendizado inicial (`LEARNING_RATE = 0.02`), δ é o fator de decaimento (`DECAY = 0.95`) e e representa o índice da época local corrente.

$$\alpha_e = \alpha_0 \cdot \delta^e \quad (3.2)$$

Ao final das 30 épocas locais, a taxa de aprendizado efetiva é:

$$\alpha_{29} = 0.02 \times 0.95^{29} \approx 0.0045 \quad (3.3)$$

```

1 # 1. Calculo dos gradientes com a penalidade da Regularizacao L2
2 np.add.at(P_grad, i_batch, errors[:, None] * Q[j_batch] - regularization * P[i_batch])
3 np.add.at(Q_grad, j_batch, errors[:, None] * P[i_batch] - regularization * Q[j_batch])
4
5 # 2. Acumulo dos gradientes passados (Momentum)
6 momentum_P = beta * momentum_P + (1 - beta) * P_grad
7 momentum_Q = beta * momentum_Q + (1 - beta) * Q_grad
8
9 # 3. Atualizacao final dos pesos
10 P += lr * momentum_P
11 Q += lr * momentum_Q

```

Algoritmo 3.1: Cálculo dos gradientes com Regularização L2 e atualização via Momentum

Os hiperparâmetros detalhados na Tabela 3.1. foram selecionados através de uma busca empírica (*grid search* manual). O critério de escolha baseou-se na calibração do ambiente federado para que este atingisse métricas de precisão equivalentes ao modelo centralizado de referência.

Parâmetro	Valor	Descrição
FACTORS	20	Número de fatores latentes (dimensão das matrizes P e Q).
BATCH_SIZE	128	Tamanho do mini-lote para o cálculo do SGD.
ITERATIONS_LOCAL	30	Número de épocas de treinamento executadas localmente por rodada.
LEARNING_RATE	0.02	Taxa de aprendizado inicial (α_0).
DECAY	0.95	Fator de decaimento da taxa de aprendizado por época.
REGULARIZATION	0.01	Termo de regularização $L2$ (λ) para evitar <i>overfitting</i> .

Tabela 3.1: Hiperparâmetros do Algoritmo de Recomendação

No contexto federado, a matriz P (fatores de usuários) foi mantida estritamente

privada no dispositivo local e reinicializada a cada rodada. Essa decisão possui duas motivações principais. Do ponto de vista de privacidade, P encapsula as preferências individuais dos usuários e, portanto, não deve ser compartilhada com o servidor. Do ponto de vista arquitetural, no framework Flower cada instância de cliente é criada sob demanda por meio da função `client_fn`, sem persistência de estado entre rodadas, o que implica que P é recalculada a cada nova convocação.

Por outro lado, apenas a matriz Q (fatores de itens) é atualizada e compartilhada com o servidor, pois representa o conhecimento coletivo sobre o catálogo de filmes, sendo adequada para agregação no processo de aprendizado federado.

3.6 ORQUESTRAÇÃO FEDERADA COM O FRAMEWORK FLOWER

A natureza dos sistemas de recomendação, onde diferentes usuários (clientes locais) interagem com o mesmo conjunto global de atributos (catálogo de filmes), caracteriza um cenário prático de aprendizado federado horizontal (Yang *et al.* (2019)). O uso dessa abordagem mitiga o problema dos silos de dados, permitindo a colaboração sem a centralização de informações sensíveis.

Para a orquestração da comunicação e gerenciamento das rodadas federadas, optou-se pela adoção do *framework* **Flower** (*flwr*) (Beutel *et al.* (2020)). Os clientes foram implementados herdando a classe `NumPyClient`, que abstrai a troca de parâmetros em formato de *arrays* multidimensionais.

A escolha do Flower em detrimento de uma orquestração puramente manual para os experimentos com criptografia justifica-se pela sua arquitetura extensível, e mais verossímil com um cenário real. Os métodos `configure_fit` e `aggregate_fit` permitem interceptar o ciclo de comunicação de forma limpa, facilitando a injeção da camada de segurança sem alterar a lógica interna de treinamento dos clientes. Contudo, cabe ressaltar que uma implementação manual *standalone* (sem *framework*) foi desenvolvida inicialmente como linha de base (*baseline*).

Para o ambiente de simulação, definiu-se um total de 50 rodadas de comunicação ($ROUNDS = 50$). Configurou-se a estratégia de agregação para selecionar um número

fixo de 10 clientes por rodada ($NUM_CLIENTS = 10$). Essa limitação fixa evitou a sobrecarga do sistema e padronizou o ruído estatístico introduzido na agregação a cada rodada.

Para garantir que os modelos locais de diferentes clientes pudessem ser matematicamente agregados pelo servidor, foi necessário padronizar a dimensionalidade das matrizes antes da aplicação da estratégia *FedAvg* McMahan *et al.* (2017). Como cada cliente interage com um subconjunto distinto do catálogo de filmes, fora desenvolvida uma rotina de alinhamento utilizando a biblioteca Pandas, conforme o Algoritmo 3.2.

```
1 aligned_Q = pd.DataFrame(local_Q, index=self.local_movie_ids) \
2     .reindex(self.global_movie_ids, fill_value=0).values
```

Algoritmo 3.2: Alinhamento do modelo local ao espaço global

O Algoritmo 3.2 acima demonstra como é feito o preenchimento com zeros ($fill_value=0$) nos filmes não avaliados pelo cliente, e assegura que a agregação no servidor ocorra apenas sobre os pesos enviados por clientes que de fato possuíam informações relevantes sobre aqueles itens específicos.

3.7 IMPLEMENTAÇÃO DA ESTRATÉGIA FEDPROX

Para mitigar a heterogeneidade dos dados inerente a sistemas de recomendação, onde usuários avaliam conjuntos de filmes completamente distintos (Non-IID), a estratégia clássica *FedAvg* foi adaptada para o algoritmo *FedProx* (Li *et al.* (2020)). O *FedProx* insere uma tolerância formal para a heterogeneidade estatística através de um termo de regularização proximal.

A implementação exigiu o envio dinâmico do hiperparâmetro μ do servidor para os clientes a cada rodada de treinamento. Nos experimentos em texto plano (`flower_fl_3_FedProx`), esse envio foi realizado por meio do *callback* `on_fit_config_fn`, registrado na estratégia *FedAvg*. Já nos experimentos com criptografia (`flower_fl_he_FedProx`), o valor de μ foi injetado diretamente no método `configure_fit` da classe *FedAvgHE*, juntamente com o contexto criptográfico público, uma vez que esse método já havia sido sobrescrito para a distribuição da chave pública. Em ambos os

cenários, o cliente recebe o valor por meio do dicionário `config` no método `fit()`, sendo acessado via `config.get("mu", 0.0)`.

```

1 proximal_term = mu * (global_Q_anchor[j_batch] - Q[j_batch])
2
3 np.add.at(Q_grad, j_batch,
4           (errors[:, None] * P[i_batch]) - (regularization * Q[j_batch]) +
           proximal_term)

```

Algoritmo 3.3: Injeção do termo proximal no gradiente de Q

O valor $\mu = 18.0$ foi determinado empiricamente durante as baterias de testes e possui respaldo teórico. Valores elevados de μ são indicados pelo próprio trabalho de Li *et al.* (2020) para cenários com alta heterogeneidade, onde a dissimilaridade entre os dados locais é acentuada. Esta é a condição que caracteriza as matrizes de recomendação estudadas neste trabalho, exigindo uma “âncora” forte para impedir que o cliente desaprenda o conhecimento global ao focar em seu nicho isolado de filmes.

3.8 CAMADA DE CRIPTOGRAFIA HOMOMÓRFICA (TENSEAL)

A garantia de confidencialidade das matrizes compartilhadas fora garantida utilizando o esquema criptográfico **CKKS**, através da biblioteca **TenSEAL**. A arquitetura de segurança segue um rigoroso fluxo de chaves assimétricas e baseia-se em um modelo de confiança onde o servidor atua estritamente como um agregador cego. Neste modelo, o servidor não possui acesso aos gradientes individuais em texto plano de nenhum cliente, sendo a sua única função somar matematicamente os vetores no domínio cifrado e descriptografar exclusivamente a média global final.

O servidor é responsável por instanciar o contexto criptográfico. A chave pública é serializada e distribuída aos clientes no início do treinamento. Os parâmetros de segurança do esquema CKKS foram definidos conforme a Tabela 3.2.

Parâmetro	Valor	Descrição
<code>poly_modulus_degree</code>	8192	Grau do polinômio, garantindo robustez de segurança.
<code>coeff_mod_bit_sizes</code>	[60, 40, 40, 60]	Tamanhos dos coeficientes do módulo para operações homomórficas.
<code>global_scale</code>	2^{40}	Escala de precisão para números reais (aprox. 12 dígitos).

Tabela 3.2: Parâmetros do Esquema CKKS

Estes parâmetros definem o nível de segurança, precisão e capacidade de computação sobre dados criptografados. A seguir, descrevemos cada um deles de forma intuitiva:

- **`poly_modulus_degree`:** Define o grau do polinômio utilizado pelo esquema criptográfico, influenciando diretamente o nível de segurança, a quantidade de dados que podem ser processados simultaneamente e o custo computacional das operações homomórficas. Em geral, valores maiores oferecem maior segurança e permitem trabalhar com cadeias de módulos mais longas, possibilitando realizar cálculos mais complexos sobre os dados criptografados. Em contrapartida, aumentam o tempo de processamento e o consumo de memória. Neste trabalho, foi utilizado o valor 8192, amplamente recomendado na literatura por proporcionar um bom equilíbrio entre segurança e desempenho.
- **`coeff_mod_bit_sizes`:** Define a cadeia de módulos utilizada pelo esquema (CKKS). Cada valor da lista representa o tamanho, em bits, de um módulo primo empregado durante as operações homomórficas. Na configuração [60, 40, 40, 60], os módulos de 60 bits localizados nas extremidades são utilizados para preservar a precisão durante a criptografia e a descriptografia, enquanto os módulos intermediários de 40 bits são consumidos pelas multiplicações homomórficas por meio da operação de reescalonamento. Dessa forma, essa configuração permite realizar aproximadamente duas multiplicações homomórficas consecutivas, além de um grande número de

operações de soma, mantendo precisão suficiente para a recuperação correta dos dados.

- **global_scale:** Controla a precisão dos números reais representados no esquema CKKS. Como o CKKS trabalha com aproximações, os valores precisam ser escalados antes da criptografia. O valor 2^{40} garante uma boa precisão numérica (aproximadamente 12 casas decimais), sendo suficiente para aplicações de aprendizado de máquina.

Em conjunto, esses parâmetros garantem que o sistema seja seguro, mantenha boa precisão nos cálculos e suporte as operações necessárias durante o treinamento federado com criptografia.

Desafio de Integração na Comunicação: O protocolo de comunicação padrão do *framework* Flower foi projetado para trafegar parâmetros na forma de tensores de ponto flutuante puros (NDArrays). Como as operações do TenSEAL geram objetos complexos do tipo *Ciphertext*, a transmissão direta causava falhas de serialização na rede.

Para solucionar esta incompatibilidade, implementamos uma rotina de conversão de baixo nível. Os vetores cifrados são serializados em sequências de *bytes* puros e, em seguida, encapsulados em matrizes de inteiros de 8 bits (`np.uint8`), conforme ilustrado no algoritmo 3.4.

```

1 enc_model = ts.ckks_vector(self.public_context, flat_q)
2 serialized_payload = enc_model.serialize()
3
4 # Reinterpretacao dos bytes cifrados como array uint8
5 # para compatibilidade com o protocolo NDArray do Flower
6 payload_array = np.frombuffer(serialized_payload, dtype=np.uint8)
7 return [payload_array], self.user_item_matrix.size, {}

```

Algoritmo 3.4: Serialização do Ciphertext para transmissão no Flower

No lado do servidor, o processo é metodicamente revertido dentro do método customizado `aggregate_fit`, onde o arranjo binário retorna ao formato de bytes brutos (`.tobytes()`) e é desserializado de volta para um objeto *Ciphertext* do TenSEAL para

permitir a soma homomórfica ponderada.

Como limitação conhecida da implementação, em caso de falha na descryptografia homomórfica — possível em situações de estouro de profundidade multiplicativa do esquema CKKS — o sistema retorna o modelo global do início do treinamento em vez dos parâmetros da rodada imediatamente anterior. Embora essa situação não tenha ocorrido nos experimentos conduzidos com os parâmetros definidos na Tabela 3.2, ela representa um ponto de atenção para implantações com maior número de rodadas ou de clientes por rodada.

Apesar da arquitetura implementada garantir a confidencialidade contra interceptações na rede, ela opera sob o modelo de segurança *Honest-but-Curious* (honesto, porém curioso) em relação ao servidor central, uma vez que este detém a chave secreta. Reconhece-se que esquemas criptográficos mais avançados, como *Multi-Key Homomorphic Encryption* (MKHE) ou a combinação de *Single-Key* com Compartilhamento de Segredos (Zhao (2022)), poderiam descentralizar essa confiança fragmentando a chave entre os clientes. Contudo, a adoção destas técnicas protocolares foram rejeitadas neste trabalho devido à sua altíssima complexidade de engenharia e ao custo computacional e de comunicação excessivo.

3.9 CENÁRIOS DE AVALIAÇÃO E MÉTRICAS

Para avaliar os *trade-offs* entre as diferentes arquiteturas propostas, os experimentos foram divididos nos seguintes cenários comparativos:

1. **Baseline de Corretude (Implementação Manual):** Simulação puramente matemática para validar a integridade da implementação do *FedAvg* antes da adoção de ferramentas externas.
2. **Baseline em Texto Plano (Flower):** Treinamento utilizando *FedAvg* e *FedProx* sem criptografia via *framework*, servindo como limite superior de precisão do modelo no ambiente final.
3. **Agregação Segura (HE):** Injeção da camada CKKS no *FedAvg* e *FedProx* para

mensurar a degradação da precisão causada pelo ruído matemático do esquema homomórfico.

A qualidade das recomendações foi mensurada utilizando as métricas clássicas (*Precisão*, *Recall* e *F1-Score*) aplicadas às dez melhores sugestões para cada usuário ($K = 10$), sendo **Precisão@10**, **Recall@10** e **F1-Score@10**.

Ao todo, foram executadas **11 configurações experimentais distintas**, resultantes da combinação de três eixos de variação: *framework* de orquestração (implementação manual vs. Flower), estratégia de agregação (FedAvg vs. FedProx com $\mu = 18.0$) e camada de privacidade (texto plano vs. criptografia homomórfica CKKS).

A Tabela 3.3 ilustra todas as configurações avaliadas.

Esquema	Framework	Estratégia	Privacidade	Clientes/Rodada
Manual_FL_sem_lim	Manual	FedAvg	Plaintext	20% (fração)
Manual_FL2	Manual	FedAvg	Plaintext	10 (fixo)
Manual_FL3_FedProx	Manual	FedProx ($\mu = 18$)	Plaintext	10 (fixo)
ManualTenseal_sem_lim	Manual	FedAvg	HE (CKKS)	20% (fração)
ManualTenseal2	Manual	FedAvg	HE (CKKS)	10 (fixo)
ManualTenseal3_FedProx	Manual	FedProx ($\mu = 18$)	HE (CKKS)	10 (fixo)
flower_fl_sem_lim	Flower	FedAvg	Plaintext	20% (fração)
flower_fl_2	Flower	FedAvg	Plaintext	10 (fixo)
flower_fl_3_FedProx	Flower	FedProx ($\mu = 18$)	Plaintext	10 (fixo)
flower_fl_he	Flower	FedAvg	HE (CKKS)	10 (fixo)
flower_fl_he_FedProx	Flower	FedProx ($\mu = 18$)	HE (CKKS)	10 (fixo)

Tabela 3.3: Configurações experimentais avaliadas

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta e analisa os resultados obtidos a partir dos experimentos detalhados na metodologia. O objetivo é avaliar a viabilidade da implementação de Sistemas de Recomendação com Aprendizado Federado Seguro, mensurando o impacto de diferentes estratégias de agregação (*FedAvg* vs. *FedProx*), a introdução da Criptografia Homomórfica (CKKS) na utilidade final do modelo, e o custo computacional associado a essas operações.

A Tabela 4.1 consolida as métricas de Precisão@10, Recall@10 e F1-Score@10 para todos os cenários executados.¹

Tabela 4.1: Resultados Consolidados das Métricas de Avaliação

Ambiente	Estratégia	Participação	Privacidade	Precisão@10	Recall@10	F1-Score@10
Manual	FedAvg	Fração (20%)	Plaintext	0.413	0.050	0.078
Manual	FedAvg	Fração (20%)	HE (CKKS)	0.413	0.050	0.078
Manual	FedAvg	10 (Fixo)	Plaintext	0.366	0.038	0.062
Manual	FedAvg	10 (Fixo)	HE (CKKS)	0.366	0.038	0.062
Manual	FedProx	10 (Fixo)	Plaintext	0.385	0.045	0.071
Manual	FedProx	10 (Fixo)	HE (CKKS)	0.385	0.045	0.071
Flower	FedAvg	Fração (20%)	Plaintext	0.403	0.048	0.075
Flower	FedAvg	10 (Fixo)	Plaintext	0.330	0.038	0.060
Flower	FedAvg	10 (Fixo)	HE (CKKS)	0.338	0.040	0.063
Flower	FedProx	10 (Fixo)	Plaintext	0.359	0.047	0.072
Flower	FedProx	10 (Fixo)	HE (CKKS)	0.366	0.048	0.073

4.1 VALIDAÇÃO DO MODELO

A primeira etapa da análise foca nos resultados gerados pela implementação manual. O propósito desta abordagem foi estabelecer uma linha de base para verificar a correteude das operações criptográficas isoladas de possíveis artefatos de rede.

Ao analisar a Tabela 4.1, constata-se um fenômeno fundamental: no ambiente manual, para uma mesma configuração de clientes e algoritmo, as métricas obtidas em texto plano

¹Nota: A precisão demonstra a capacidade de assertividade do modelo de sugerir boas recomendações sobre todas as recomendações feitas aos usuários, sendo considerado verdadeiro positivo (VP), somente quando aquele item fora avaliado com nota maior, ou igual a 3 (nota ≥ 3). Destaca-se que a inclusão de HE (CKKS) não degradou a utilidade do modelo e que a participação por fração (20%) otimizou os resultados comparado à participação fixa.

(*Plaintext*) e com criptografia homomórfica (*HE CKKS*) são rigorosamente idênticas. Os cenários *Manual FedAvg (Fração e 10 Fixos)* e *Manual FedProx (10 fixos)*, por exemplo, obtiveram F1-Score idênticos tanto com os dados abertos quanto cifrados.

Este resultado comprova que a parametrização do esquema CKKS escolhida (grau polinomial $N = 8192$ e escala global de 2^{40}) foi bem dimensionada. O esquema suportou nativamente a aritmética sobre números reais sem que o truncamento homomórfico corrompesse os gradientes.

Adicionalmente, ao comparar os dois ambientes de simulação, observa-se que a orquestração via framework Flower introduz uma sutil degradação nas métricas em relação à implementação manual. Por exemplo, a Precisão@10 no cenário FedAvg (Fração) recua de 0,413 no manual para 0,403 no framework. Essa flutuação não decorre de falhas no modelo, mas do overhead sistêmico de um ambiente distribuído, que envolve processos assíncronos de serialização e desserialização de parâmetros para o protocolo de rede. Essa distinção de baseline reforça a importância de ter estabelecido o ambiente manual como uma prova de conceito para validar a integridade dos algoritmos antes da migração para uma arquitetura de orquestração mais complexa.

4.2 O IMPACTO DA HETEROGENEIDADE E A EFICÁCIA DO FEDPROX

Em sistemas de recomendação, o catálogo de itens varia drasticamente de usuário para usuário, caracterizando um cenário de dados estritamente *Non-IID* (não-independentes e não-identicamente distribuídos).

No ambiente orquestrado pelo *Flower*, quando a participação de clientes sofreu restrição para um limite rígido de 10 clientes fixos sob a estratégia *FedAvg*, o F1-Score retraiu de forma acentuada em relação ao cenário fracionário (de 0.075 para 0.060). Esse comportamento é sintomático do *client drift* com poucos participantes, onde os modelos locais otimizam-se excessivamente em direção às preferências de nicho dos usuários daquela rodada específica, “esquecendo” o ótimo global. Foi necessário utilizar um número fixo de clientes pois ao acrescentar a criptografia junto ao flower, atingimos um limiar máximo de memória no ambiente utilizado.

Ficou provada a eficácia do algoritmo *FedProx* na mitigação deste problema. Ao substituir o *FedAvg* pelo *FedProx* no cenário do Flower com 10 clientes, o F1-Score saltou para 0.072 (um ganho relativo de 20%), e a Precisão elevou-se de 0.330 para 0.359, com o treinamento realizado em cima de textos planos. Com a implementação da HE o F1-Score saltou para 0.073 (um ganho relativo de aproximadamente 16%) e a Precisão elevou-se de 0.338 para 0.366. O termo de regularização proximal ($\mu = 18.0$) forçou os clientes a extraírem seus padrões locais sem se desviarem agressivamente das características globais estabelecidas pelo servidor. O melhor termo proximal foi encontrado através do uso de um algoritmo de *gridsearch* que compara o resultado das métricas.

4.3 ORQUESTRAÇÃO SEGURA E O RUÍDO COMO REGULARIZADOR IMPLÍCITO

Nos experimentos orquestrados pelo *Flower*, nota-se um fenômeno contra-intuitivo e de alto interesse científico: a introdução da criptografia não apenas manteve a precisão do modelo, como provocou uma **ligeira melhora** nas métricas de avaliação. No cenário *Flower FedProx*, a Precisão subiu de 0.359 em texto plano para 0.366 em ambiente cifrado (HE). Da mesma forma, no cenário *Flower FedAvg*, a precisão subiu de 0.330 para 0.338.

Essa flutuação positiva não configura uma melhoria intencional gerada pelo esquema CKKS, mas é o reflexo de um conceito clássico de aprendizado de máquina: o ruído como **regularizador implícito**. O esquema CKKS, por definição, insere um pequeno ruído matemático em seus coeficientes para garantir a segurança da cifra contra ataques quânticos, além de realizar truncamentos numéricos aproximados durante a serialização (`uint8`) e desserialização via rede.

No contexto de Fatoração de Matrizes, que atua sobre representações hiper-esparsas e sofre cronicamente com *overfitting*, a injeção desse ruído em escala microscópica durante a agregação dos pesos previne que o modelo decore os dados de treinamento de forma rígida. Consequentemente, o modelo global resultante apresenta uma generalização sutilmente superior ao ser avaliado nos dados de teste, validando que a privacidade via HE não apenas é viável, mas pode convergir de forma mais robusta.

4.4 AVALIAÇÃO DO TEMPO DE EXECUÇÃO

Embora as métricas de utilidade atestem o sucesso da arquitetura, a viabilidade prática de sistemas federados depende diretamente do custo computacional (*overhead*). A Tabela 4.2 apresenta o tempo total de treinamento registrado para as 50 rodadas em cada experimento.

Tabela 4.2: Tempos Totais de Treinamento (50 Rodadas Federadas)

Ambiente	Estratégia	Participação	Privacidade	Tempo (s)	Tempo (min)
Manual	FedAvg	Fração	Plaintext	43.36	0.72
Manual	FedAvg	Fração	HE (CKKS)	2542.01	42.37
Manual	FedAvg	10 (Fixo)	Plaintext	8.44	0.14
Manual	FedAvg	10 (Fixo)	HE (CKKS)	383.79	6.40
Manual	FedProx	10 (Fixo)	Plaintext	4.30	0.07
Manual	FedProx	10 (Fixo)	HE (CKKS)	227.43	3.79
Flower	FedAvg	Fração	Plaintext	1044.52	17.41
Flower	FedAvg	10 (Fixo)	Plaintext	952.04	15.87
Flower	FedAvg	10 (Fixo)	HE (CKKS)	1033.95	17.23
Flower	FedProx	10 (Fixo)	Plaintext	843.87	14.06
Flower	FedProx	10 (Fixo)	HE (CKKS)	1017.68	16.96

A análise revelou disparidades cruciais na escalabilidade do sistema. Observando isoladamente o ambiente **Manual**, a introdução da camada TenSEAL no cenário *FedAvg* (10 fixos) inflacionou o tempo de processamento de forma severa, saltando de 8.44 segundos para 383.79 segundos (um acréscimo absoluto de ≈ 375 segundos). Como esta implementação executa os clientes de forma sequencial, este intervalo quantifica o custo puro exigido pelas rotinas de encriptação polinomial, soma homomórfica e decriptação no servidor.

No ambiente **Flower**, a dinâmica de tempo é alterada devido à arquitetura concorrente do *framework*. O custo de orquestração que envolve serialização assíncrona, trocas de mensagens e alocação de memória, fizeram o tempo do *FedAvg Plaintext* (10 fixos) elevar para 952 segundos. Ao adicionar a camada HE, o tempo saltou para 1034 segundos, sendo um acréscimo de apenas ≈ 82 segundos. O fato deste acréscimo ser muito inferior ao medido no cenário manual explica-se pela capacidade do *framework* de paralelizar o

processamento local dos clientes em múltiplos núcleos da CPU, mitigando o impacto do tempo de criptografia no tempo total de execução.

Apesar do ganho conferido pelo paralelismo, a soma dos tempos demonstram que o volume massivo dos vetores expandidos pelo esquema CKKS e a carga computacional das operações homomórficas configuram obstáculos substanciais.

Estes resultados concluem que, embora a Criptografia Homomórfica seja perfeitamente capaz de manter a eficácia de recomendação, o volume massivo dos vetores expandidos pelo esquema CKKS e a carga computacional das operações no domínio cifrado configuram os maiores obstáculos para a implantação na nuvem em tempo real. Esta barreira de desempenho expõe a fronteira da pesquisa atual, servindo de base fundamentada para futuras investigações e projetos de otimização arquitetural.

5 CONCLUSÕES E TRABALHOS FUTUROS

5.1 CONCLUSÕES GERAIS

O presente trabalho teve como objetivo investigar a viabilidade técnica e os impactos na precisão ao integrar Sistemas de Recomendação, Aprendizado Federado e Criptografia Homomórfica. A motivação central focou na necessidade de construir sistemas capazes de prover filtragem colaborativa eficiente sem violar a privacidade dos usuários ou centralizar dados sensíveis.

Os resultados obtidos confirmaram as hipóteses iniciais. Através da implementação de uma arquitetura orquestrada pelo *framework Flower*, provou-se ser possível manter os vetores de características dos usuários (matriz P) estritamente locais, trafegando apenas o conhecimento coletivo sobre os itens (matriz Q). A introdução do algoritmo *FedProx* demonstrou-se fundamental e eficaz para lidar com o viés estatístico inerente aos dados de recomendação (cenários *Non-IID*), mitigando o *client drift* e elevando o F1-Score e a Precisão de forma consistente em redes com restrição de participação de clientes.

No que diz respeito à segurança, a incorporação do esquema criptográfico CKKS, via biblioteca *TenSEAL*, atingiu o objetivo de converter o servidor em um agregador cego. A análise das métricas de recuperação de informação revelou a descoberta mais importante desta pesquisa, onde as aproximações numéricas exigidas pela aritmética homomórfica sobre números reais não degradaram a utilidade preditiva do modelo. Pelo contrário, o micro-ruído introduzido atuou como um leve regularizador implícito, preservando integralmente a qualidade das recomendações entregues ao usuário final.

Conclui-se, portanto, que a arquitetura proposta atende aos preceitos de *Privacy by Design*, sendo funcionalmente viável para resguardar a privacidade em sistemas de recomendação baseados em Fatoração de Matrizes.

5.2 LIMITAÇÕES E TRABALHOS FUTUROS

Apesar do sucesso na manutenção da precisão preditiva, os experimentos evidenciaram que a garantia matemática da privacidade impõe um severo custo na camada de engenharia.

A conversão de matrizes de parâmetros em polinômios cifrados pelo esquema CKKS gerou um aumento expressivo no tempo de processamento das instâncias locais e do servidor, além de inflacionar de forma massiva o tamanho das mensagens (*payloads*) trafegadas na rede.

Esta constatação revela que implementações ingênuas de Criptografia Homomórfica em ambientes federados podem tornar o sistema inviável para aplicações em tempo real devido à saturação da largura de banda e à latência computacional.

Identifica-se, portanto, uma lacuna crítica na implementação eficiente de alternativas descentralizadas que garantam confidencialidade total sem comprometer a escalabilidade do sistema. Diante desses desafios, sugere-se a continuidade desta pesquisa através das seguintes frentes de investigação:

- **Métricas de Eficiência de Rede:** Quantificar formalmente a expansão de dados e o volume de tráfego gerado pelos vetores criptografados, visando identificar gargalos de largura de banda em redes convencionais.
- **Otimização de Parâmetros de Segurança:** Realizar análises variando os parâmetros do esquema CKKS, como o grau polinomial ($N = 4096$ e $N = 8192$) e os níveis de bits dos coeficientes, para encontrar o equilíbrio ideal entre robustez criptográfica e latência de processamento.
- **Análise de Convergência em Alta Heterogeneidade:** Avaliar o desempenho de estratégias de agregação robustas sob condições extremas de dados *Non-IID*, investigando se o ruído de aproximação do CKKS interfere na estabilidade do modelo em larga escala.
- **Redução do Overhead Computacional:** Explorar técnicas de compressão ou arquiteturas de agregação segura que minimizem o custo da CPU no servidor durante a soma homomórfica dos pesos.

O aprofundamento nessas variáveis permitirá o fornecimento de diretrizes de engenharia fundamentais para que sistemas de recomendação com preservação de privacidade possam ser aplicados em ambientes de produção de forma eficiente e escalável.

REFERÊNCIAS

- Batista, J. P. C., Sarmiento, E. M., Bastos, J. J., Mota, V. F., Villaça, R. S., **Mitigação de Ataques “Label-Flipping” no Aprendizado Federado: Experimentos e Estratégias de Seleção de Clientes**. In: Escola Regional de Informática do Espírito Santo (ERI-ES), 2024. SBC, pp. 91–98.
- Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., de Gusmão, P. P., Lane, N. D., 2020. **Flower: A friendly federated learning research framework**. arXiv preprint arXiv:2007.14390.
- Cheon, J. H., Kim, A., Kim, M., Song, Y., **Homomorphic encryption for arithmetic of approximate numbers**. In: International conference on the theory and application of cryptology and information security, 2017. Springer, pp. 409–437.
- Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT press.
- Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V., 2020. **Federated optimization in heterogeneous networks**. Proceedings of Machine learning and systems 2, 429–450.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B. A., **Communication-Efficient Learning of Deep Networks from Decentralized Data**. In: Artificial intelligence and statistics, 2017. PMLR, pp. 1273–1282.
- Motta, C. D., Garcia, A., Vivacqua, A., Santoro, F. M., Sampaio, J., 2011. **Sistemas de recomendação**. Pimentel, M.; Fuks, H. “Sistemas colaborativos”. Rio de Janeiro: Elsevier.
- Orlandi, F. C., 2023. **Mitigando o impacto de dados non-IID em federated learning com entropia**.
- Ruder, S., 2016. **An overview of gradient descent optimization algorithms**. arXiv preprint arXiv:1609.04747.
- Vinko, D., Köhler, M., Miličević, K., Lukić, I., **Application of Homomorphic Encryption for a Secure-by-Design Approach to Protect the Confidentiality of Data in Proficiency Testing and Interlaboratory Comparisons**. In: Telecom, 2026. Vol. 7. MDPI, p. 14.
- Yang, Q., Liu, Y., Chen, T., Tong, Y., 2019. **Federated machine learning: Concept and applications**. ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2), 1–19.
- Zhang, Y., 2022. **An introduction to matrix factorization and factorization machines in recommendation system, and beyond**. arXiv preprint arXiv:2203.11026.
- Zhao, D., 2022. **Skefl: Single-key homomorphic encryption for secure federated learning**. arXiv preprint arXiv:2212.11394.