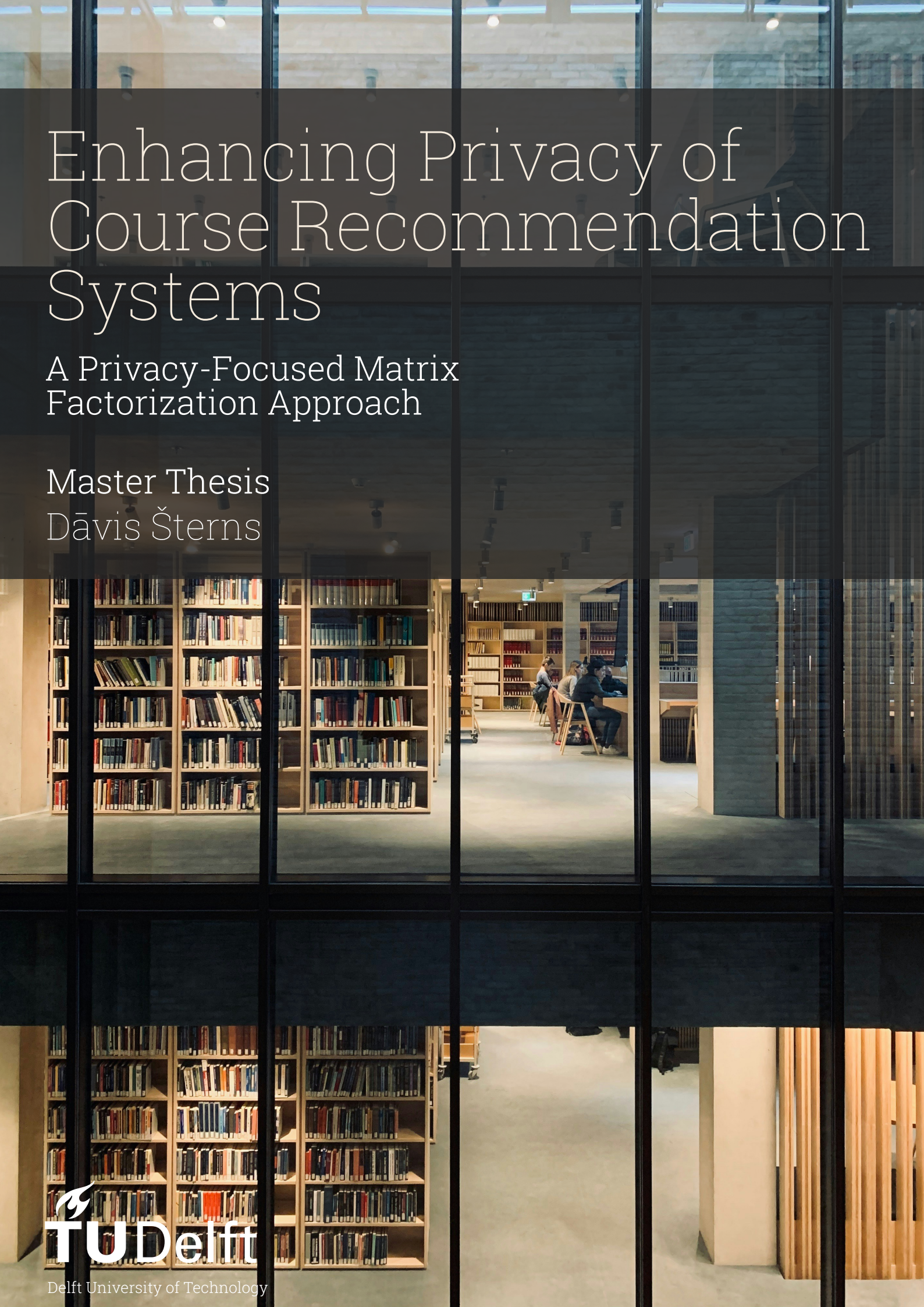

Enhancing Privacy of Course Recommendation Systems

A Privacy-Focused Matrix Factorization Approach

Master Thesis

Dāvis Šterns

TU Delft
Delft University of Technology

Enhancing Privacy of Course Recommendation Systems

A Privacy-Focused Matrix Factorization Approach

by

Dāvis Šterns

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 21, 2025 at 15:00.

Supervisor:	Z. Erkin
Daily Supervisor:	F.W. Dekker
Project Duration:	January, 2024 - August, 2025
Faculty:	Faculty Electrical Engineering, Mathematics and Computer Science, Delft

Cover: Budapest, Central European University. Photo by Mark Kakas on Unsplash

TU Delft

Abstract

Personalized course-recommendation systems can help students make better academic choices and improve learning outcomes. Matrix factorization (MF) is a well-established and effective approach for this task, producing accurate recommendations from historical student–course performance data. However, the deployment of MF-based recommenders is hindered by privacy and regulatory risks, particularly when sensitive student records are processed by third-party or centralized systems. In the privacy-preserving setting, MF models exhibit reduced accuracy: when combined with differential privacy, accuracy is fundamentally degraded by the added noise, while existing cryptography-based approaches omit bias terms, resulting in a measurable accuracy gap with their plaintext equivalents.

This thesis enhances a Homomorphic-Encryption-based recommendation protocol to support biased Matrix Factorization through two additions: data centering and vector augmentation. These modifications maintain the security guarantees of the original protocol under the semi-honest adversary model while enabling the model to incorporate user and item biases. Evaluated in the plaintext domain on the MovieLens-100k dataset, the enhanced model achieved a test RMSE of 0.9213, a notable improvement over the baseline's 0.9507, and reached the baseline's best RMSE with only 15 training iterations instead of 145. Beyond accuracy and efficiency, separating bias terms from the student–course interaction extends the system from a simple grade predictor into a tool for academic discovery, allowing for recommendations that consider inherent compatibility, not solely predicted grades. Although demonstrated in a course-recommendation setting, the approach is applicable to any privacy-preserving recommender system, offering reduced computational costs and narrowing the accuracy gap with non-private methods.

Preface

It is done.

One afternoon after a Security and Cryptography lecture, I gathered my courage, walked up to Professor **Zeki**, and asked, “I think I like cryptography. Can I do my thesis with you?” He smiled and replied, “Make sure you take the PETs course and we can arrange something.”

I later learned that PETs stood for Privacy-Enhancing Technologies and, sadly, had nothing to do with cute animals. Still, the course became my gateway into this thesis. Overall, Zeki’s guidance was subtle — it never felt like explicit learning, yet looking back, I realize I’ve learned far more than I noticed at the time. A great supervisor knows how to work with students from very different backgrounds and mindsets, finding ways to make each of them tick. Zeki, I truly appreciate the way you did that for me.

Of course, it isn’t just my supervisor whom I want to thank. This thesis is the result of a beautiful, messy, tangled network of signals and support from the people around me, starting with my thesis committee. I am grateful to **Masoud** for his careful evaluation of this work and for a valuable discussion on recommender systems earlier that helped me better understand the topic.

It was also great having wise and skilled PhD students around in the office, always available for a quick chat and a cup of tea. A very special thanks to **Florine**, my daily supervisor and a constant inspiration. Whenever I was stuck, I’d think, “What would Florine do in this situation?” You have a way of untangling any complex idea and seeing it clearly. **Jelle**, your knowledge of homomorphic encryption is incredible, and you’re a phenomenal presenter. **Jorrit**, I truly admire your conviction that privacy is something worth fighting for. **Tjitkse**, thanks for all the delicious home-baked muffins and for always being such fun and cheerful company. And to all the other people in the **Cybersecurity Research Group**: I think there is a part of each of you that I have absorbed and will always carry with me (and yes, that includes you, **Sandra**).

Among my fellow Master’s students, I especially want to thank **Chelsea**, **Prakhar**, **Marco**, and **Rutger** for being there for me, for your support when it wasn’t easy, and for being such great friends! Overall, moving to a different country for my Master’s degree has been a pretty fun experience, and I am very grateful for all the connections and friends made along the way. **Çağrı**, **Doeke**, **Willemijn**, **Emile**, **Niko**, **Iván**, and so many more of you. You know who you are, and I’m grateful for every one of you!

Among these wonderful people, I feel especially fortunate to have met **Torrin**. You’ve been right beside me through it all—a great listener, a constant supporter... well, you are simply a part of my life, and I am immensely grateful for that.

And although it can be difficult to keep in touch with friends who live far away, I am very grateful to have you all, especially those in the “**Massiv Hank**” and “**Citronskābe?**” group chats. You are my backbone.

Also, thanks for the support from my colleagues at the **Electronic Communications Office of Latvia**, who were very understanding and accommodating when my mind was busy with this thesis.

Frankly, I also owe thanks to the **AI tools** that I used as brainstorming partners, text editors, and occasional therapists when I stared at a blank page for too long.

I want to close this section with immense gratitude to my family, who have always supported me, encouraged me to go and explore the world, and were always there on our weekly Thursday video calls! I have made it this far because of you. **Emīl**, **Matīs**, **Mom**, and **Dad**, I love you all.

Now, I think I will go outside and take a small break before my next academic adventures in Helsinki.

Dāvis Šterns
Delft, August 2025

Contents

1 Introduction

The transition from secondary to higher education is a critical point in a student's life. The courses they choose at this stage can define their academic journey and shape their future career prospects [1]. Choosing the right courses from brief descriptions, among different specializations and peer suggestions, is a daunting task. Poor course choices can lead to a cascade of negative outcomes, including student demotivation, course withdrawals, extended study periods, and, in some cases, dropping out of university altogether [2]. These consequences represent not only a significant loss of time and monetary resources for the individual student, but also reflect a systemic inefficiency within educational institutions, resulting in the misallocation of staff and funding. The traditional course advising, such as one-on-one meetings with a study advisor, offers personalization, however, it is often time-consuming and difficult to scale, leaving many students inadequately supported [3].

1.1. Course Recommendation Systems and Student Privacy

As higher education increasingly embraces digitalization, accelerated by the global shift to remote and hybrid learning, universities are collecting unprecedented amounts of student data [4]. This data holds the promise of revolutionizing student support through automated, personalized course recommendation systems. However, this vast collection of sensitive information creates a high-risk environment. Student data - containing grades, academic performance history, high school records, and even demographic information - is deeply personal [5]. Universities often outsource the development and hosting of these digital services to third-party vendors, creating a landscape where data security is not guaranteed [6]. This was infamously demonstrated by Edmodo, an educational data technology provider that used children's personal data for advertising without explicit consent [7].

Furthermore, this sensitive information can be leaked through data breaches or hacked by malicious actors. Documented cases show the scope of this harm: in one instance, a single student was able to download and publicly share an unprotected file from a university server, exposing the personal details and grades of 5,962 other students [8]. In a more tragic case, criminals, who had purchased a student's stolen university application data used it to impersonate education officials. After successfully scamming the student out of her entire tuition savings, the acute stress and despair from the fraud triggered a fatal cardiac arrest [9]. Even with the best intentions, a service provider remains a critical point of vulnerability; the data can be exposed through unintentional leaks or be repurposed should the company be acquired by an entity with conflicting data protection guidelines [10].

This highlights the urgent need for course recommendation systems that protect student privacy by design, not merely by policy [11] - instead of relying on the service provider to keep the information secure, the system itself should be architecturally incapable of exposing raw student data. This imperative leads to the central research question of this thesis:

Research Question

How can we design a course recommendation system that protects student privacy by keeping their grades hidden from the service provider and any other third parties, while still providing accurate and personalized recommendations?

1.2. Selecting an Algorithmic Foundation

To answer this question, we must first select an appropriate algorithmic foundation. A review of course recommendation systems by Guruge, Kadel, and Halder [12] identifies three dominant course recommendation approaches: Content-based, Collaborative Filtering, and Knowledge-Based.

Of these, Knowledge-Based (KB) systems pose a serious ethical problem. These systems operate by reasoning over a knowledge base of explicit rules. While some rules are benign, such as ”students interested in programming should take Python,” the core issue arises when incorporating student demographic information. From a purely predictive standpoint, this approach can seem deceptively powerful. Indeed, multiple studies have shown correlations between demographics and academic pathways. For instance, Charles and Bradley [13] demonstrated that the enrollment in various courses is influenced by gender. Furthermore, Carnevale and Strohl [14] showed that socioeconomic status, often linked to race and income, can correlate with performance and persistence in certain demanding academic programs. In a KB system these statistical patterns could be easily turned into rules, potentially increasing the raw accuracy of its recommendations. However, this is precisely where the ethical danger lies. Using such data risks reinforcing societal biases and promoting systemic discrimination. For example, by observing that students from low-income backgrounds have historically enrolled less in advanced mathematics, the system might actively steer new students from similar backgrounds away from these courses, regardless of their individual academic potential. This creates a deterministic feedback loop that perpetuates inequality, making KB systems ethically unfit for our purposes.

Content-based recommendations, another common approach, also present significant limitations for our purpose. These systems suggest courses based on item features, recommending courses similar to those a student has already taken or rated highly. For instance, if a student has succeeded in ”Introduction to Java,” a content-based model would recommend ”Advanced Java” or ”Object-Oriented Programming.” The drawbacks of this approach are as follows: first, they require existing preference data to function, making them ineffective for new students. Second, and more importantly, they fail to utilize the rich, historical performance data available across the entire student body. By focusing only on an individual’s past, they miss the opportunity to learn from the successes and failures of thousands of other students. This narrow focus also means they struggle to recommend novel or diverse options, effectively trapping students in an ‘echo chamber’ of their initial interests and limiting their academic exploration.

In contrast to the limitations of the aforementioned approaches, Collaborative Filtering (CF) emerges as the most promising paradigm for our purposes. It can sidestep the ethical dilemmas of Knowledge-Based systems by purely operating on the obtained final grades rather than sensitive demographic information. Recommendations are therefore based on demonstrated academic merit, not on preconceived notions of what a student ’should’ study. Furthermore, it overcomes the narrow ’echo chamber’ of Content-Based methods by leveraging the ’wisdom of the crowd’ - the collective academic histories of the entire student body. This allows the system to identify non-obvious yet successful academic pathways by finding students with similar performance profiles, offering a richer and more personalized form of guidance. Nevertheless, a critical challenge for CF is a ’cold start’ problem - meaning that first-year students lack prior data in the system to obtain useful, personalized recommendations for their initial courses. To overcome this, we can incorporate high school academic records. Research by Danilowicz-Gösele, Lerche, Meya, *et al.* [15] has shown that high school GPA is among the best predictors of university success; we hypothesize that incorporating the full subject grade distribution, not just the average, could provide an even richer data source for initializing the recommendation model. The advantages and disadvantages of each major approach discussed are summarized in Table 1.1.

With a strategy to address new students, the next critical decision lies in selecting the specific CF im-

Table 1.1: Overview of pros and cons of major recommendation system approaches.

Knowledge-Based (KB) Systems	
Pros	**Cons**
• Transparent decision rules • Can encode domain knowledge	• High ethical risk from demographic bias • Can reinforce societal inequality
Content-Based Systems	
Pros	**Cons**
• Personalized to student's interests • Avoids demographic data	• Echo chamber effect • Ignores wider student data
Collaborative Filtering (CF)	
Pros	**Cons**
• Avoids demographics • Leverages collective information • More diverse recommendations	• Cold start problem

plementation. The field is broadly divided into two families: neighborhood-based and model-based methods. Neighborhood-based approaches operate on a principle of direct similarity, recommending courses that have been successful for a small group of 'neighboring' students with the most similar academic histories. In contrast, model-based methods attempt to learn the underlying latent factors — the hidden characteristics of both students and courses — that explain the observed performance data. While neighborhood methods are intuitive, model-based approaches have proven to be more powerful in uncovering complex patterns in student performance data. Work by Thanh-Nhan, Nguyen, and Thai-Nghe [16] demonstrates that a model-based approach, specifically Matrix Factorization (MF), significantly outperforms neighborhood methods in terms of recommendation accuracy and computational performance. They further identify biased Matrix Factorization as the superior variant. This improved approach explicitly models the global mean and biases - systematic tendencies observed in the data. For instance, some students might consistently achieve higher grades than others (represented as user bias). Similarly, some courses are intrinsically harder than others (represented as item bias). By isolating these systematic tendencies, the core model can focus on the more nuanced interactions, leading to more accurate predictions [17]. Therefore, we select biased Matrix Factorization as the algorithmic core of our system.

1.3. Towards Private and Accurate Recommendations

Having chosen our engine, the next challenge is to make it private. This introduces a fundamental tension between three competing goals: Accuracy, Privacy, and Computational Complexity. Data anonymization, a common first step, has been proven insufficient; as Narayanan and Shmatikov [18] notoriously demonstrated, individuals can be re-identified in anonymized datasets with minimal auxiliary information. A more robust technique is Differential Privacy (DP), which adds controlled noise to data or model outputs to provide formal privacy guarantees [19]. However, DP introduces an inherent trade-off: stronger privacy requires more noise, which degrades the accuracy of the recommendations [20]. Federated Learning (FL) offers a complementary approach where, instead of sending raw data to a central server, each user trains a local model and only submits model updates (e.g., gradients) for aggregation [21]. While this reduces data exposure, Li, Ding, Zhang, *et al.* [22] have shown that model updates in MF can still be used to reconstruct user input data, meaning FL often requires DP to be truly secure, thereby reintroducing the accuracy-privacy trade-off.

Cryptographic methods, particularly those using homomorphic encryption (HE), offer a way to resolve this tension. HE allows for computations to be performed directly on encrypted data, enabling strong privacy guarantees without sacrificing model accuracy. Protocols developed by Nikolaenko, Ioannidis,

Weinsberg, *et al.* [23], later optimized with data-packing techniques by Kim, Kim, Koo, *et al.* [24], have demonstrated the feasibility of fully cryptographic matrix factorization. While these methods introduce substantial computational complexity, they are unique in their ability to provide privacy without compromising the quality of the result. This guarantee of privacy without sacrificing recommendation accuracy is the reason for our inclusion of HE as the preferred encryption method within the proposed algorithm.

1.4. Contributions

To answer our research question, this thesis puts forward a novel system that combines homomorphic encryption with biased Matrix Factorization, thereby addressing a critical gap in the research: the absence of privacy-preserving matrix factorization algorithms that incorporate user, item, and global mean biases. This is not a minor omission. In the non-private recommender systems literature, these bias terms are widely used to account for individual user (student) or item (course) variations independent of their interactions, allowing the MF model to pay closer attention to the more nuanced user-item interaction. Furthermore, our preliminary experiments indicate that these terms also lead to significantly faster model convergence during training. The primary technical goal of this thesis is to integrate these crucial components into a private MF protocol. This approach creates a more accurate system, closing the performance gap between private and non-private recommenders. Moreover, the methodology we develop for incorporating these biases is designed to be efficient and could serve as a blueprint for future privacy-preserving matrix factorization algorithms.

This thesis makes the following contributions:

- **Systematization and Formalization of a Foundational Protocol:** We present a thorough explanation of the privacy-preserving matrix factorization protocol by Kim, Kim, Koo, *et al.* [24]. Our work resolves ambiguities and formalizes underspecified components from the original publication, yielding the first rigorous and complete algorithmic description required for implementation and further extensions.
- **Proposal of a Novel Improvement and Adaptation:** We architect a complete course recommendation system based on this algorithm and propose a novel improvement by integrating user, item, and global mean biases into the cryptographic model. This extension is designed to significantly increase recommendation accuracy and improve computational performance during model training.
- **Empirical Validation and Performance Analysis:** We conduct a two-part evaluation. First, we provide a theoretical analysis of our method's modest computational overhead, which, as we later show, is far outweighed by the accuracy improvements. Second, we conduct empirical experiments in the plaintext domain to validate the improvements. By comparing a standard (plaintext) matrix factorization algorithm with and without bias terms, we empirically validate the necessity of our approach. Since our chosen cryptographic protocol does not introduce noise, we argue that these plaintext performance results carry over directly to the cryptographic domain. This analysis shows that critical accuracy improvements are achieved with negligible additional complexity, thus validating the practicality and effectiveness of our proposed system.

1.5. Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces the technical preliminaries of recommender systems and cryptographic primitives. Chapter 3 reviews related work. Chapter 4 provides a detailed analysis of the baseline protocol by Kim et al. In Chapter 5, we present our core contribution: the design of our Privacy-Preserving Course Recommender System, detailing both its adaptation for this specific use case and our novel improvements to incorporate biases. Chapter 6, details the experiments and evaluation of our system's security and performance. Finally, Chapter 7 discusses the implications of our findings and outlines directions for future work.

2

Technical Background

The previous chapter motivated the use of biased Matrix Factorization and homomorphic encryption, which serve as the foundation of this thesis. This chapter provides the necessary technical background on these two topics.

2.1. Recommender Systems

Recommender systems are computational tools designed to combat information overload by predicting a user's preference for items within some catalog. They provide personalized suggestions by analyzing past user-item interactions or item attributes. This work focuses on the dominant paradigm known as Collaborative Filtering, which leverages historical user behavior to generate recommendations.

2.1.1. Collaborative Filtering Recommender Systems

Collaborative Filtering (CF) is a paradigm of recommendation that operates on a single, powerful assumption: users who have rated items similarly in the past are likely to give similar ratings in the future. The system begins with a set of known ratings, where user i has given a rating r_{ij} to item j. The goal is to analyze these known ratings to recommend items that the user hasn't rated but would likely have a high preference for.

Unlike content-based methods that analyze item characteristics, CF relies entirely on the user-item interaction data. Within CF, two main families of algorithms exist:

- **Neighborhood-based methods** are memory-based techniques that make predictions by identifying similar entities. In user-based approaches, predictions for a user are generated by finding a "neighborhood" of other users with similar rating histories and recommending items they have liked. In item-based approaches, the similarity is calculated between items instead of users, and predictions are made by suggesting items that are similarly rated to those a user has already rated highly.
- **Model-based methods** use known ratings to learn a mathematical model that represents the underlying latent structure of the data. This model is then used to make predictions.

Model-based approaches, and specifically Matrix Factorization, have been shown to be effective at capturing the complex patterns in sparse user-item data and often yield superior accuracy [25]. The remainder of this section will focus on this technique.

2.1.2. Matrix Factorization Models

Matrix Factorization is a model-based, pointwise recommender system. The term "pointwise" means that the system functions by estimating a potential rating for each user-item pair individually. Based on these estimations, it can then present each user with a ranked list of items they are most likely to prefer.

The model works under the assumption that user preferences and item attributes can be described by

unobserved, or latent "profiles." Each user i is associated with a profile vector $\mathbf{u}_i \in \mathbb{R}^d$, and each item j with a profile vector $\mathbf{v}_j \in \mathbb{R}^d$. The hyperparameter d defines the dimensionality of this latent space. The model then assumes that user i's rating for an item j can be estimated by the interaction between their respective profiles, which is modeled as the inner product of their vectors:

$$\hat{r}_{ij} = \mathbf{u}_i^T \mathbf{v}_j. \tag{2.1}$$

The goal of Matrix Factorization is to learn the optimal profile vectors for all users and items given the set of existing ratings. This is achieved by finding the profiles that make the predicted ratings $\hat{r}_{ij}$ as close as possible to the known ratings r_{ij}. This task is formulated as an optimization problem that minimizes the sum of squared errors between predictions and actual ratings. To prevent the model from overfitting to the training data, ℓ_2 regularization terms are added, which penalize large profile vectors.

Table 2.1 defines the mathematical notation used in the following sections.

Table 2.1: A description of the symbols used in the model.

Symbol	Description	Type / Domain
n, m	The number of users and items, respectively	$\mathbb{N}^+$
$\mathcal{M}$	The set of user-item pairs (i, j) for which a rating is known	$\subset \{1, \dots, n\} \times \{1, \dots, m\}$
r_{ij}	The known rating given by user i to item j	$\mathbb{R}$
$\hat{r}_{ij}$	The predicted rating for user i on item j	$\mathbb{R}$
d	The dimensionality of the latent feature space	$\mathbb{N}^+$
$\mathbf{u}_i$	The latent feature vector (profile) for user i	$\mathbb{R}^d$
$\mathbf{v}_j$	The latent feature vector (profile) for item j	$\mathbb{R}^d$
μ	The global average of all known ratings in $\mathcal{M}$	$\mathbb{R}$
b_i	The bias term for user i	$\mathbb{R}$
b_j	The bias term for item j	$\mathbb{R}$
λ	The regularization hyperparameter	$\mathbb{R}^+$
γ	The learning rate for gradient descent	$\mathbb{R}^+$

Standard Matrix Factorization

The complete objective function $\mathcal{L}$ for standard Matrix Factorization is:

$$\min_{\mathbf{u}_*, \mathbf{v}_*} \mathcal{L} = \sum_{(i,j)\in\mathcal{M}} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda \left(\sum_{i=1}^{n} \|\mathbf{u}_i\|^2 + \sum_{j=1}^{m} \|\mathbf{v}_j\|^2 \right), \tag{2.2}$$

where λ is a hyperparameter controlling the strength of the regularization.

The method's name comes from an alternative linear algebra perspective. The known ratings can be organized into a sparse user-item matrix $\mathbf{R} \in \mathbb{R}^{n\times m}$. The goal is then to find two low-rank matrices, a user-feature matrix $\mathbf{U} \in \mathbb{R}^{n\times d}$ (where row i is $\mathbf{u}_i^T$) and an item-feature matrix $\mathbf{V} \in \mathbb{R}^{m\times d}$ (where row j is $\mathbf{v}_j^T$), whose product approximates $\mathbf{R}$. The predicted rating matrix, $\hat{\mathbf{R}} = \mathbf{U}\mathbf{V}^T$, 'completes' the sparse matrix, providing a prediction for every user-item pair. This formulation is equivalent to the pointwise model, as the entry $(\hat{\mathbf{R}})_{ij}$ is precisely the dot product $\mathbf{u}_i^T \mathbf{v}_j$. This perspective is useful as it connects the problem to the well-studied area of low-rank matrix completion, allowing for the application of tailored algorithms and optimized computational methods.

Biased Matrix Factorization

A powerful extension to the standard model involves incorporating bias terms to account for systematic tendencies in rating data. In the context of course recommendations, some students (users) may generally achieve higher grades, and some courses (items) may be inherently easier. Biased Matrix Factorization explicitly models these effects.

The prediction rule is augmented with three bias terms:

$$\hat{r}_{ij} = \mu + b_i + b_j + \mathbf{u}_i^T \mathbf{v}_j. \tag{2.3}$$

Here, μ is the global average rating across all $\{r_{ij} : (i,j) \in \mathcal{M}\}$, b_i is a user-specific bias, and b_j is an item-specific bias. The profile interaction $\mathbf{u}_i^T \mathbf{v}_j$ now only needs to capture the user-item interaction beyond these general effects. The corresponding objective function is updated to include these new parameters:

$$\min_{\mathbf{u}_*, \mathbf{v}_*, b_*} \mathcal{L} = \sum_{(i,j)\in\mathcal{M}} (r_{ij} - (\mu + b_i + b_j + \mathbf{u}_i^T \mathbf{v}_j))^2 + \lambda \left(\sum_{i=1}^{n} \|\mathbf{u}_i\|^2 + \sum_{j=1}^{m} \|\mathbf{v}_j\|^2 + \sum_{i=1}^{n} b_i^2 + \sum_{j=1}^{m} b_j^2 \right). \tag{2.4}$$

2.1.3. Model Optimization via Gradient Descent

The objective function in Equation 2.4 is non-convex, meaning an optimal solution cannot be found with a direct analytical method. Instead, iterative optimization algorithms are employed to find a locally optimal set of parameters. While several methods exist, such as Stochastic Gradient Descent (SGD) and Alternating Least Squares (ALS), this work utilizes Batch Gradient Descent.

In Batch Gradient Descent, the optimization proceeds in epochs, where each epoch involves a full pass over the training set $\mathcal{M}$. For each epoch, the gradient of the loss function $\mathcal{L}$ with respect to each parameter is computed by summing the contributions from every known rating $(i,j) \in \mathcal{M}$. Let the prediction error be $e_{ij} = r_{ij} - \hat{r}_{ij}$. The partial derivatives of the loss function $\mathcal{L}$ are:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}_i} = \sum_{j:(i,j)\in\mathcal{M}} -2e_{ij}\mathbf{v}_j + 2\lambda \mathbf{u}_i \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{v}_j} = \sum_{i:(i,j)\in\mathcal{M}} -2e_{ij}\mathbf{u}_i + 2\lambda \mathbf{v}_j \tag{2.5}$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \sum_{j:(i,j)\in\mathcal{M}} -2e_{ij} + 2\lambda b_i \qquad \frac{\partial \mathcal{L}}{\partial b_j} = \sum_{i:(i,j)\in\mathcal{M}} -2e_{ij} + 2\lambda b_j \tag{2.6}$$

After computing these gradients for all parameters, they are updated simultaneously by taking a single step in the opposite direction of the gradient, scaled by the learning rate γ. The update rules are as follows, where the factor of 2 from the derivative is absorbed into γ for notational simplicity:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \gamma \left(\lambda \mathbf{u}_i - \sum_{j:(i,j)\in\mathcal{M}} e_{ij}\mathbf{v}_j \right) \qquad \forall i \in \{1,\dots,n\}, \tag{2.7}$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j - \gamma \left(\lambda \mathbf{v}_j - \sum_{i:(i,j)\in\mathcal{M}} e_{ij}\mathbf{u}_i \right) \qquad \forall j \in \{1,\dots,m\}, \tag{2.8}$$

$$b_i \leftarrow b_i - \gamma \left(\lambda b_i - \sum_{j:(i,j)\in\mathcal{M}} e_{ij} \right) \qquad \forall i \in \{1,\dots,n\}, \tag{2.9}$$

$$b_j \leftarrow b_j - \gamma \left(\lambda b_j - \sum_{i:(i,j)\in\mathcal{M}} e_{ij} \right) \qquad \forall j \in \{1,\dots,m\}. \tag{2.10}$$

This process is repeated over multiple epochs until the parameters converge or a stopping criterion is met.

2.1.4. Evaluation Metrics

To assess model performance, a portion of the data is held out as a test set, $\mathcal{T}$. The model is trained on the remaining data and evaluated on this unseen test set. We use two standard metrics that measure distinct qualities of a recommender system.

Root Mean Squared Error (RMSE)

RMSE measures the accuracy of the model's rating predictions. It represents the standard deviation of the prediction errors, providing a measure of the average magnitude of error. A lower RMSE indicates higher predictive accuracy. It is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} (r_{ij} - \hat{r}_{ij})^2}. \tag{2.11}$$

Normalized Discounted Cumulative Gain (nDCG@k)

While RMSE measures prediction accuracy, it fails to evaluate the quality of a ranked recommendation list, which is more important in the course recommendation scenario, as a student is unlikely to consider a course ranked far down the list. We therefore need a metric that rewards placing the most relevant items at the top. The Normalized Discounted Cumulative Gain (nDCG) is the standard for this task.

The logic of nDCG is best understood with an example. Assume our system recommends 5 courses ($k = 5$), and we know the student's actual grades for them, which serve as the true relevance scores.

Table 2.2: An example of a recommended course list and the ground truth relevance (actual grades) used to calculate nDCG@5.

Position (j)	Recommended Course	Actual Grade (rel_j)	Ideal Ordering
1	Course C	7.0	Course A (9.5)
2	Course A	9.5	Course E (8.0)
3	Course D	4.0	Course C (7.0)
4	Course B	2.0	Course D (4.0)
5	Course E	8.0	Course B (2.0)

First, we calculate the **Discounted Cumulative Gain (DCG)**. This metric scores the list by summing the relevance (gain) of each item, but penalizes items based on their rank. The relevance of an item at position j is divided by a logarithmically growing term, heavily "discounting" items further down the list.

$$\text{DCG@}k = \sum_{j=1}^{k} \frac{\text{rel}_j}{\log_2(j+1)}. \tag{2.12}$$

For the recommended list in Table 2.2, the DCG@5 is approximately 19.0.

However, this raw score is not easily comparable across different users. To create a fair score between 0.0 and 1.0, we normalize it. We calculate the DCG of the best possible ranking, shown as the "Ideal Ordering" in the table. This is the **Ideal Discounted Cumulative Gain (IDCG)**. For the example, the IDCG@5 is approximately 20.5. The final nDCG score is the ratio of the actual DCG to the ideal DCG:

$$\text{nDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}. \tag{2.13}$$

In our example, the score is $19.0/20.5 \approx 0.92$. This means the system's list achieved 92% of the quality of a perfect ranking. An nDCG score of 1.0 represents the ideal ranking. As will be discussed in Chapter 6, this metric is particularly well-suited for the course recommendation task.

2.2. Privacy-Preserving Techniques

As established in the introduction, the vast collection of sensitive student data necessitates a system that protects privacy by design, not merely by policy. This goal requires the use of Privacy-Enhancing Technologies (PETs), a field of techniques designed to protect personal data while still allowing for useful computation. The European Union's General Data Protection Regulation (GDPR) has formalized principles like data minimization, making the application of PETs a legal and ethical imperative [26].

Broadly, PETs can be divided into two paradigms, which differ fundamentally in their approach to the inherent tension between privacy and utility. The first approach involves altering the data itself to obscure

identifying information, while the second uses cryptography to shield the data, allowing for computation without ever exposing it. This section will review both approaches to build the foundation for our selection of a privacy-preserving framework.

2.2.1. Approaches Based on Data Alteration and Noise

The most intuitive approach to data privacy is to modify the dataset before it is shared or analyzed, with the goal of making it impossible to link records back to specific individuals. These techniques operate on a spectrum of increasing complexity and robustness.

The most straightforward method, **basic anonymization**, involves stripping explicit identifiers like names and social security numbers from a dataset. However, this was famously shown to be insufficient by Sweeney [27], who demonstrated that 87% of the U.S. population can be uniquely identified by their 5-digit ZIP code, gender, and full date of birth (known as quasi-identifiers). To counter this, a family of more advanced statistical techniques was developed. **k-Anonymity** requires that each record in a dataset be indistinguishable from at least $k-1$ other records along its quasi-identifiers, typically by generalizing specific values into ranges [28]. While an improvement, k-Anonymity is vulnerable to attacks if the sensitive attributes within an indistinguishable group lack variety. For instance, if a k-anonymous group of patients all share the same diagnosis, an adversary knows the diagnosis of everyone in that group. This led to further refinements like **l-diversity**, which mandates a minimum level of diversity for sensitive attributes within each group [29], and **t-closeness**, which requires that the distribution of a sensitive attribute within a group be close to its distribution in the overall dataset [30].

While the aforementioned methods protect against specific attack models, **Differential Privacy (DP)** represents a paradigm shift by offering a formal, mathematical definition of privacy [19]. Rather than modifying the dataset's structure, DP is a property of a query or algorithm. It ensures that the output of an analysis will be statistically indistinguishable whether any single individual's data is included in the dataset. This is achieved by injecting carefully calibrated statistical noise into the results. The level of privacy is governed by a parameter, ϵ (epsilon); smaller values of ϵ provide stronger privacy guarantees but require the addition of more noise. This creates a fundamental and inescapable privacy-accuracy trade-off: stronger privacy guarantees directly lead to less accurate results, while higher accuracy can only be achieved by weakening the privacy guarantee.

2.2.2. Cryptography-Based Approaches

An alternative paradigm resolves the privacy-accuracy trade-off by leaving the data entirely intact. Rather than altering data or adding noise to results, cryptographic methods enable computations to be performed directly on unchanged encrypted data. While this avoids adding statistical noise to results, it introduces a significant computational overhead, creating its own trade-off between privacy and performance.

Encryption

The goal of any encryption scheme is to transform a readable message, known as a plaintext (denoted by m), into an unreadable ciphertext (denoted by c). This transformation is performed by an encryption algorithm (enc), and the reverse process is handled by a decryption algorithm (dec). The security of these operations relies on a piece of secret information known as a key. At the core of modern cryptography are two fundamental approaches to encryption.

The first approach, **symmetric encryption**, uses a single, shared secret key (denoted by k) for both encrypting and decrypting a message. Formally, this is expressed as $c = enc_{\mathsf{k}}(m)$ for encryption, and $m = dec_{\mathsf{k}}(c)$ for decryption. The primary advantage of this method is its computational speed. However, its main drawback is the need for a secure channel through which all communicating parties can first agree on the shared key k.

The second approach, **asymmetric encryption** (also known as public-key cryptography), was introduced by Diffie and Hellman [31] to solve this key distribution problem. It uses a pair of mathematically linked keys: a public key (pk) that can be shared openly with anyone, and a corresponding secret key (sk) that is kept secret by its owner. The public key is used for encryption ($c = enc_{\mathsf{pk}}(m)$), but only the holder of the secret key can perform the decryption ($m = dec_{\mathsf{sk}}(c)$). This eliminates the need for a pre-shared secret, as anyone can encrypt a message for the intended recipient using their public key,

confident that only the recipient can decrypt it.

Homomorphic Encryption

Standard encryption schemes are rigid; once data is encrypted, it cannot be meaningfully manipulated. **Homomorphic Encryption (HE)** overcomes this limitation by enabling computation directly on encrypted data [32]. An encryption scheme is considered homomorphic for a given operation if it is possible to process ciphertexts in a way that obtains a new ciphertext that, when decrypted, matches the result of applying that same operation to the original plaintexts. For example, an encryption scheme is considered additively homomorphic if for two messages m_1 and m_2, their sum $(m_1 + m_2)$ can be directly computed given their individual encryptions:

$$enc_{\mathsf{pk}}(m_1) \oplus enc_{\mathsf{pk}}(m_2) = enc_{\mathsf{pk}}(m_1 + m_2), \tag{2.14}$$

where $\oplus$ represents the homomorphic addition operation performed on the ciphertexts.

A prominent example is the Paillier cryptosystem, whose security relies on the difficulty of factoring large numbers [33]. Key generation involves selecting two large primes p and q, computing $n = p \cdot q$ as the public modulus, and choosing a base g (commonly $n+1$). The public key is (n, g), while the secret key consists of the prime factors p and q.

The Paillier cryptosystem operates on messages from the message space $\mathbb{Z}_n$. To encrypt a message $m < n$, a random value $r \in \mathbb{Z}_{n^2}^*$ is chosen and the ciphertext is computed as $c = g^m \cdot r^n \pmod{n^2}$. Decryption computes:

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n \tag{2.15}$$

where $\lambda = \mathrm{lcm}(p-1, q-1)$ is the least common multiple of $(p1)$ and $(q1)$, and the function $L(x)$ is defined as $L(x) = (x1)/n$.

The generalization of this concept is **Fully Homomorphic Encryption (FHE)**, supporting both addition and multiplication operations on encrypted data. This capability is powerful, as it enables the evaluation of any arbitrary function on a ciphertext [34], making it theoretically possible to securely outsource any computation to an untrusted environment.

The first fully homomorphic encryption (FHE) scheme, a significant theoretical breakthrough, was introduced by Gentry in 2009 and was based on ideal lattices [34]. Despite its importance, this initial construction proved too inefficient for practical applications. A major advancement toward practical FHE came in 2012 with schemes founded on the Ring Learning With Errors (RLWE) problem [35]. By operating over polynomial rings, these schemes achieved substantial efficiency gains [36].

Subsequent research has yielded several significant RLWE-based variants. The Brakerski-Gentry-Vaikuntanathan (BGV) [37] and Brakerski/Fan-Vercauteren (BFV) [36], [38] schemes are particularly well-suited for exact integer arithmetic. In contrast, the Cheon-Kim-Kim-Song (CKKS) scheme is optimized for approximate arithmetic on real or complex numbers, making it highly effective for applications such as machine learning [39]. More recent developments, including Torus FHE (TFHE), have focused on further optimizing performance for specific operations like bootstrapping [40]. This ongoing research continues to enhance the practicality of FHE for deployment in real-world systems.

Data Packing

A key challenge in using HE is that security parameters often require very large ciphertexts, regardless of the size of the underlying message. Encrypting every 32-bit value in a separate ciphertext would be highly inefficient. Data packing is a critical optimization that addresses this by encoding multiple plaintext values into a single large ciphertext. In RLWE-based FHE, this is achieved through the polynomial structure: the coefficients of a single plaintext polynomial $(m_0, m_1, \ldots, m_{N-1})$ can represent a vector of many individual messages. Homomorphic operations (like addition or multiplication) then act on all these messages simultaneously in a SIMD (Single Instruction, Multiple Data) fashion. For matrix factorization protocols, as demonstrated by Kim, Kim, Koo, *et al.* [24], this technique allows an entire vector of user ratings or item features to be packed into one ciphertext, dramatically reducing the number of ciphertexts and the overall computational complexity.

2.2.3. Secure Computation Outsourcing via Data Masking

While FHE is theoretically capable of any computation, some operations remain prohibitively expensive or complex to execute homomorphically. In such cases, a hybrid approach combining HE with a secure interactive protocol can be employed. This allows a client to securely "outsource" a difficult computation to a server holding the secret key, without revealing any underlying data.

Consider a scenario involving two parties:

- **Party $\mathcal{A}$ (Data Owner)** has the encryption of the original sensitive data x, denoted by $enc_{\mathsf{pk}}(x)$, which is encrypted under the HE public key pk. Party $\mathcal{A}$ wants to obtain $enc_{\mathsf{pk}}(y)$, where $y = f(x)$ for some function f. Performing f homomorphically is computationally expensive.
- **Party $\mathcal{B}$ (Computation Server):** Possesses the HE secret key sk corresponding to pk, and thus can perform decryption $dec_{\mathsf{sk}}(\cdot)$. Party $\mathcal{B}$ has the computational resources to execute the function f efficiently on plaintext data. Party $\mathcal{A}$ trusts $\mathcal{B}$ to execute f correctly but does not want $\mathcal{B}$ to learn the value of x.

The goal of the interaction is for Party $\mathcal{A}$ to obtain an encryption of the result, $enc_{\mathsf{pk}}(y)$ (with $y = f(x)$), using Party $\mathcal{B}$'s ability to compute f in plaintext, while ensuring that Party $\mathcal{B}$ never gains knowledge of the original data x.

The following procedure is detailed for an additive HE scheme. However, the underlying principle of this data masking interaction is more general: it is applicable with any HE scheme provided that the employed masking strategy ensures the target function f can be decomposed. This decomposability allows the data owner to separate the effect of the function on the original data from its effect on the mask (see more at item 7 - unmasking step).

1. **Masking (Party $\mathcal{A}$):** Party $\mathcal{A}$ generates a secret random mask σ from an appropriate domain. Then, using the homomorphic properties of the HE scheme, $\mathcal{A}$ computes a masked ciphertext

$$enc_{\mathsf{pk}}(x^*) := enc_{\mathsf{pk}}(x) \oplus enc_{\mathsf{pk}}(\sigma) = enc_{\mathsf{pk}}(x + \sigma) \tag{2.16}$$

2. **Send to Server (Party $\mathcal{A} \to \mathcal{B}$):** Party $\mathcal{A}$ sends the masked ciphertext $enc_{\mathsf{pk}}(x^*)$ to Party $\mathcal{B}$.
3. **Decrypt Masked Data (Party $\mathcal{B}$):** Party $\mathcal{B}$ uses its secret key sk to decrypt the received ciphertext, obtaining

$$x^* = dec_{\mathsf{sk}}(enc_{\mathsf{pk}}(x^*)) = dec_{\mathsf{sk}}(enc_{\mathsf{pk}}(x + \sigma)) = x + \sigma. \tag{2.17}$$

   Crucially, since Party $\mathcal{B}$ does not know the random mask σ chosen solely by Party $\mathcal{A}$, it cannot obtain the original value x from the decrypted masked value $x^* = x + \sigma$.
4. **Compute on Masked Data (Party $\mathcal{B}$):** Party $\mathcal{B}$ applies the desired function f to the plaintext masked data:

$$y^* := f(x^*) = f(x + \sigma). \tag{2.18}$$

5. **Re-encrypt Result (Party $\mathcal{B}$):** Party $\mathcal{B}$ encrypts the result using the public key pk, obtaining $enc_{\mathsf{pk}}(y^*)$.
6. **Send Result Back (Party $\mathcal{B} \to \mathcal{A}$):** Party $\mathcal{B}$ sends $enc_{\mathsf{pk}}(y^*)$ to Party $\mathcal{A}$.
7. **Unmasking (Party $\mathcal{A}$):** Party $\mathcal{A}$ receives $enc_{\mathsf{pk}}(y^*)$. To obtain the desired result $enc_{\mathsf{pk}}(y)$—an encryption of $y = f(x)$—the effect of the mask σ introduced in Step 1 needs to be removed. This step requires that this effect in f is separable from the result of the function without knowing the secret value x. In other words, $f(x + \sigma)$ should be decomposable as:

$$f(x + \sigma) = f(x) + f_\sigma, \tag{2.19}$$

   where f_σ is a function of σ that does not depend on x, and this decomposition should be known to Party $\mathcal{A}$, meaning that they have a method to determine the value of f_σ. For instance, if f is an affine function such that $f(x) = ax + b$, the decomposition is straightforward: $f(x + \sigma) = a(x + \sigma) + b = (ax + b) + a\sigma = f(x) + f_\sigma$, where Party $\mathcal{A}$ can easily compute the mask's effect $f_\sigma = a\sigma$ since it knows both the function parameter a and its own secret mask σ. This

decomposition is not possible for all functions. For example, a non-linear function like $f(x) = x^2$ results in $f(x+\sigma) = (x+\sigma)^2 = x^2 + 2x\sigma + \sigma^2$. Since Party $\mathcal{A}$ cannot compute the cross-term $2x\sigma$ without knowing x, this function cannot be evaluated with this simple additive masking protocol.

Then the desired value of $enc_{\mathsf{pk}}(f(x))$ can be obtained by homomorphically adding the negation of f_σ to the received $enc_{\mathsf{pk}}(y^*)$:

$$\begin{aligned} enc_{\mathsf{pk}}(y^*) \oplus enc_{\mathsf{pk}}(-f_\sigma) &= enc_{\mathsf{pk}}(f(x+\sigma)) \oplus enc_{\mathsf{pk}}(-f_\sigma) \\ &= enc_{\mathsf{pk}}(f(x) + f_\sigma) \oplus enc_{\mathsf{pk}}(-f_\sigma) \\ &= enc_{\mathsf{pk}}(f(x) + f_\sigma - f_\sigma) \\ &= enc_{\mathsf{pk}}(f(x)) \\ &= enc_{\mathsf{pk}}(y) \end{aligned} \tag{2.20}$$

2.2.4. Fixed-Point Arithmetic

Homomorphic encryption schemes operate on plaintext spaces of integers (e.g., $\mathbb{Z}_N$ or polynomial rings), whereas the matrix factorization model requires arithmetic on real numbers. To resolve this, we use fixed-point arithmetic, which represents real numbers as scaled integers. A real number x is encoded as $\overline{x}$ by expressing it in a binary representation, scaling it with a precision factor 2^α, and taking the floor:

$$\overline{x} = \lfloor x \cdot 2^\alpha \rfloor \tag{2.21}$$

This is analogous to representing monetary values like €1.23 as 123 cents to avoid decimal points. To recover the approximate real value, the process is reversed: $x \approx \overline{x}/2^\alpha$.

Arithmetic on these encoded values requires careful management of the scale. Addition and subtraction are direct, as the scale is preserved: $\overline{x} \pm \overline{y} \approx \overline{x \pm y}$. However, multiplication and division alter the scale.

When multiplying two fixed-point numbers, their scales multiply, requiring a correction.

$$\overline{x} \cdot \overline{y} \approx (x \cdot 2^\alpha) \cdot (y \cdot 2^\alpha) = (x \cdot y) \cdot 2^{2\alpha} \tag{2.22}$$

To return to the original scale of 2^α, the result must be divided by the scaling factor. This is known as rescaling.

$$\overline{x \cdot y} = \left\lfloor \frac{\overline{x} \cdot \overline{y}}{2^\alpha} \right\rfloor \tag{2.23}$$

This rescaling step via division inherently involves a truncation or rounding, which can introduce a small precision error in the computation.

In contrast, division requires pre-scaling to maintain precision. To compute $\overline{x/y}$, the numerator ($\overline{x}$) must be scaled by 2^α before being divided by the encoded denominator ($\overline{y}$). This ensures both the proper scaling and the numerical accuracy of the division result.

$$\overline{x/y} \approx \left\lfloor \frac{x}{y} \cdot 2^\alpha \right\rfloor = \left\lfloor \frac{(x \cdot 2^\alpha)}{y \cdot 2^\alpha} \cdot 2^\alpha \right\rfloor \approx \left\lfloor \frac{\overline{x}}{\overline{y}} \cdot 2^\alpha \right\rfloor \tag{2.24}$$

Thus, computations in the encrypted domain must carefully track and adjust the scaling factor to ensure the correctness of the final result. The choice of α balances precision with the computational constraints of the cryptosystem.

3

Related Work

3.1. Privacy-Preserving Recommender Systems

Privacy concerns in recommender systems have become a critical research focus as these systems increasingly rely on sensitive user data to generate personalized recommendations. Even anonymized rating datasets can be re-identified by linking them with auxiliary information [18], while temporal changes in public outputs may reveal user transactions Calandrino, Kilzer, Narayanan, *et al.* [41]. Moreover, matrix factorization (MF) representations can leak private attributes through inference on user/item vectors, even without direct access to raw ratings Resheff, Elazar, Shahar, *et al.* [42]. In response, the field has evolved from ad hoc perturbation techniques to formal differential privacy (DP), and—when the server is untrusted—to local DP and cryptographic approaches.

3.1.1. Randomization-Based Approaches

Early work by Polat and Wenliang Du [43] proposed adding randomized noise to user ratings before sharing them with the service. This demonstrated the fundamental utility-privacy trade-off: higher noise protects privacy but hurts recommendation quality (shown on neighborhood-based CF). This approach was later formalized with the introduction of differential privacy [44].

Central differential privacy (trusted server). In the central model, a trusted server collects raw ratings and adds calibrated noise during the aggregation/learning phase. A notable study by McSherry and Mironov [20] adapted both neighborhood (item-item) and factor (SVD-like) approaches by injecting noise into the final covariance and weight matrices. The released statistics and models then satisfy ε-DP (Laplace) or (ε, δ)-DP (Gaussian) under standard composition. This protects against inferences from the published outputs while requiring trust in the server to hold raw data. This approach additionally introduces an accuracy–privacy trade-off relative to plaintext training McSherry and Mironov [20].

Local differential privacy (untrusted server). When the server cannot be trusted, the ratings submitted by the users should be protected. Shin, Kim, Shin, *et al.* [45] applied local DP to MF: each user adds noise to their contributions before the upload. Local DP strengthens the threat model but typically incurs larger utility loss; dimensionality reduction and tailored noise partially mitigate this gap [45].

Despite these advancements in models and mechanisms, a fundamental tension persists: enhancing privacy through the injection of noise impacts the precision of the underlying data and, consequently, the recommendation accuracy. Therefore, the core challenge in this domain remains the effective management of this inherent privacy-utility trade-off.

3.1.2. Federated Learning Approaches

The emergence of federated learning has introduced alternative paradigms for privacy-preserving recommender systems. Chai, Wang, Chen, *et al.* [46] and Ammad-ud-din, Ivannikova, Khan, *et al.* [47] explored federated matrix factorization approaches where users maintain local models and only share

model updates. However, [48], [49] and [50] demonstrated various attacks that could recover original data by observing exposed model updates, requiring additional privacy protection mechanisms. Xu, Chu, and Song [51] demonstrated that this attack could be prevented by adding differential privacy mechanisms that protect exposed information. Although accuracy is better than simply using DP, this approach reintroduces the privacy-utility trade-off that federated learning approaches sought to eliminate.

3.1.3. Cryptography-Based Approaches

While FL reduces reliance on a central data holder, it still leaves model updates vulnerable to inference. Cryptographic approaches instead aim to remove trust in the server entirely by ensuring that all computations occur over encrypted data. Within this field, research has generally followed two distinct tracks: securing neighborhood-based models and securing matrix factorization models.

One major line of research has focused on applying cryptography to neighborhood-based collaborative filtering. Foundational work by Canny [52], [53] utilized homomorphic encryption for this purpose, but these early protocols incurred prohibitive computational and communication overhead. Erkin et al. later introduced more efficient versions [54], [55], although they initially relied on active user participation. A subsequent, more robust design eliminated the need for user-provider communication, albeit at the cost of slower performance [56]. This area continues to be an active field of research, as the relative simplicity of neighborhood-based calculations makes them a practical target for newer and more efficient cryptographic primitives.

A separate research thrust has aimed to secure matrix factorization (MF) models, which are known for their high accuracy in non-private settings [25]. Nikolaenko et al. [23] pioneered this direction with a protocol combining garbled circuits and homomorphic encryption. While comprehensive, its immense computational cost made it impractical. Kim et al. [24] achieved a critical 50-fold performance increase through innovative data packing and the use of a non-colluding crypto-service provider. However, despite these improvements, the intrinsic complexity and high overhead of secure matrix factorization have remained significant challenges, leading to less research activity in this specific area in recent years. Furthermore, a notable research gap exists within these protocols, as to date, none incorporate user biases or the global mean—standard components for maximizing the accuracy of their non-private counterparts.

3.2. Privacy in the Educational Context

The deployment of data-driven systems in education introduces a unique set of challenges that go beyond the technical implementation of recommendation algorithms. Experts have identified “data privacy” as the most critical factor for establishing trust and quality in Learning Analytics [57].

3.2.1. The Digital Revolution in Education: A Double-Edged Sword

Modern higher education has undergone a significant digital transformation, creating a rich ecosystem of Educational Technology (EdTech) that manages nearly every aspect of academic life [5]. This digitalization has opened the door for institutions to take advantage of large amounts of student data, such as predicting academic performance [58], identifying students at risk of dropping out [59], predicting academic behavior [60] and even detecting cheating in online assessments [61], [62].

However, this increased data collection creates a double-edged sword. The complexity of the EdTech ecosystem, which often involves numerous third-party vendors, introduces significant security vulnerabilities. High-profile data breaches, such as the compromise of the MOVEit file transfer tool that affected hundreds of schools, have exposed the sensitive personal data of countless students, including names, dates of birth, and even Social Security numbers [63]. The consequences of such breaches can be devastating for the individuals involved [9]. This establishes a clear and present danger that motivates the search for more secure data-handling paradigms.

3.2.2. Privacy Harms

Reidenberg and Schaub [64] identify harms arising from excessive information collection and processing, which can lead to student identification, profiling, and surveillance. Even though the technology

might provide additional support for students, it might increase the performance-related stress if not implemented appropriately [64].

A common defense offered by institutions is the anonymization of datasets before processing or release. However, research has demonstrated that anonymization is an insufficient safeguard. For instance, Yacobson, Hershkovitz, Fuhrman, *et al.* [65] showed that patterns in de-identified student log data could be used to re-identify physical classes and schools with the help of publicly available information. Similarly, Chen, Davis, Lin, *et al.* [66] were able to successfully re-identify 42% of learners in a dataset by linking it with data from social media. These studies prove that simply stripping direct identifiers from a dataset does not guarantee privacy, necessitating a move toward more robust technical protections.

3.2.3. Obligation to Know vs. The Right to Privacy

The challenge of student privacy is not merely technical but is rooted in a deep ethical conflict. On one hand, institutions may have an *obligation of knowing* [67]. This perspective argues that it could be considered unethical *not* to use available data to intervene and support a student who is on a likely path to academic failure. This duty of care creates a powerful incentive to analyze student data.

On the other hand, this obligation clashes directly with fundamental principles of student privacy and autonomy [68], [69]. The core of this conflict is captured by a series of unresolved questions regarding data ownership, access rights, and security [70]. This is even more relevant when processing student demographic information. Research by Paquette, Li, Baker, *et al.* [71] found that 15% of recent publications in educational data mining used demographic variables. While this can improve predictive accuracy, it risks introducing and amplifying societal biases, potentially steering students away from certain pathways based on their background rather than their individual merit.

3.2.4. Mitigation Strategies

In response to these challenges, researchers and practitioners have proposed various mitigation strategies, which can be broadly categorized as policy-based and data-level interventions.

Policy-based approaches focus on establishing frameworks for trust and governance. For example, Drachsler and Greller [72] provided an eight-point checklist for trusted Learning Analytics, while Reidenberg and Schaub [64] proposed policy recommendations for building privacy and accountability into educational technologies. While essential, these frameworks are not technical guarantees; their effectiveness hinges on the faithful and correct implementation by all parties, ultimately relying on a model of trust.

Data-level interventions attempt to apply technical fixes directly to the data. Many researchers advocate for a strict data-minimization principle [5], [73]. Others have proposed technical redaction methods to automatically remove private information from unstructured data like forum posts [74]. The limitation of these approaches is that they often force a direct trade-off between privacy and utility; minimizing or redacting data inherently reduces its richness and can degrade the performance of the models that rely on it.

3.2.5. The Need for Privacy by Design

The existing landscape of solutions reveals a critical gap. Policy-based solutions rely on trust, which can be breached, while data-level interventions often sacrifice model accuracy for privacy. This thesis argues that this trade-off is a limitation of the current toolkit and can be avoided. The path forward is to embrace the principle of *Privacy by Design* [75], embedding privacy protections directly into the architecture of the system itself.

This leads directly to the cryptographic approaches at the core of this thesis. By leveraging techniques like homomorphic encryption, as explored in the work of Kim, Kim, Koo, *et al.* [24] and extended here, it becomes possible to perform complex computations—such as training a matrix factorization model—directly on encrypted data. This paradigm is fundamentally different: it does not require trusting the service provider with raw data, nor does it require degrading the data's integrity. It resolves the core conflict by enabling the use of student data for their benefit while ensuring their privacy is mathematically guaranteed, not just policy-protected.

4

Privacy-Preserving Matrix Factorization by Kim et al.

We describe a privacy-preserving matrix factorization algorithm proposed by Kim, Kim, Koo, *et al.* [24]. This is a cryptography-based algorithm for performing matrix factorization while preserving user privacy. This research addressed computational bottlenecks associated with cryptographic primitives and improves on a privacy-preserving matrix factorization algorithm proposed by Nikolaenko, Ioannidis, Weinsberg, *et al.* [23]. The primary contributions are a novel cryptographic protocol that introduces an additional, non-colluding party—the Crypto-Service-Provider (CSP)—and a ”Data Packing” method that combines multiple computations into a single cryptographic operation.

The core idea of the algorithm is to allow the Recommender System (RecSys) to learn the user and item profiles $\mathbf{u}_i$ and $\mathbf{v}_j$ for each user i and each item j, so that the prediction of the rating that user i would give to item j, $\hat{r}_{ij}$ can be estimated as $\mathbf{u}_i^T \mathbf{v}_j$. Unlike the standard matrix factorization, which is explained in section 2.1, all computations, including user ratings and profile vectors, are performed on encrypted data, preventing RecSys from directly accessing this information. The authors note that this could be achieved by naively replacing every arithmetic operation in the standard matrix factorization algorithm by its homomorphic equivalent. However, this would be very inefficient.

This protocol differentiates itself from this naive approach through two key innovations with the aim of improving efficiency:

1. Data packing: instead of performing all computations independently, combining multiple variables into the same array and performing computations on the array increases the efficiency.
2. CSP interaction with data masking: RecSys masks sensitive data before sending it to the CSP. The CSP can then operate on this masked data in the plaintext domain, which is significantly more efficient than performing the equivalent operations on encrypted data.

With these improvements, the authors report a speed improvement of 50 times over Nikolaenko, Ioannidis, Weinsberg, *et al.* [23], who used Garbled Circuits [76] and Additive Homomorphic Encryption to perform privacy-preserving matrix factorization.

4.1. Nomenclature

The protocol described in this chapter employs a range of variables, from standard matrix factorization parameters to specialized data structures for cryptographic computation. To ensure clarity, this section provides a systematic overview of the notation used.

4.1.1. Notational Conventions

The notation follows a set of conventions to distinguish between different forms of a variable:

- **Typeface:** Scalars and set-related identifiers are written in a normal font (e.g., r_{ij}, M), while vectors are denoted by boldface letters (e.g., $\mathbf{u}_i$, $\mathbf{U}$).
- **Fixed-Point Representation ($\overline{x}$):** As cryptographic operations are performed on integers, real numbers are converted to a fixed-point representation. A variable with a bar, such as $\overline{x}$, denotes the fixed-point version of the real number x. With a precision parameter α, this is defined as $\overline{x} = \lfloor x \cdot 2^{\alpha} \rfloor$. For brevity, the bar may be omitted where the context makes the representation clear.
- **Packed Vectors ($\mathbf{U}, \mathbf{V}, \ldots$):** To optimize performance, the algorithm "packs" multiple individual data items (like all user profile vectors associated with the ratings in $\mathcal{M}$) into single, large vectors. These are denoted by bold, uppercase letters like $\mathbf{U}$, $\mathbf{V}$, $\hat{\mathbf{U}}$, and $\hat{\mathbf{V}}$. Their precise structure is detailed in section 4.2.
- **Masked Variables (x^*):** To securely delegate computation, RecSys masks data before sending it to the CSP.
  - A variable in a masked state is denoted with a '*' superscript (e.g., x^*).
  - The random value used for masking is generally denoted by σ.
  - After the CSP performs computations on a masked variable, the term representing the effect of the random value on the mask is denoted by f_σ.

  This notation represents the core principle of the data masking interaction detailed in subsection 2.2.3.
- **Iteration-Dependent Variables ($X(k)$):** The Learning Phase is an iterative process. Variables that are updated in each iteration (or epoch) are indexed by the iteration number k in parentheses, such as $\mathbf{U}(k)$, which represents the user profile matrix at the end of iteration k. The initial state is denoted by $k = 0$.
- **Encryption and Decryption:** The general form for encryption is $enc_{\mathsf{pk}}(m)$, signifying the encryption of message m with public key pk. For simplicity, the shorthands $enc(\cdot)$ and $dec(\cdot)$ are used to denote encryption with the FHE public key and decryption with the FHE secret key, respectively. The specific keys and functions are detailed in the table below.
- **Element-Wise Multiplication ($\odot$):** The symbol $\odot$ denotes the element-wise (Hadamard) product. For two vectors $\mathbf{a}$ and $\mathbf{b}$ of the same dimension, the resulting vector $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$ is defined by $c_i = a_i b_i$.

4.1.2. Table of Variables

The following table lists all variables used in the protocol, grouped by their function.

Symbol	Description	Type / Domain
	1. General and System Parameters	
n, m	The total number of users and items.	$\mathbb{N}$
d	The dimension of the latent feature vectors.	$\mathbb{N}$
$[n], [m]$	The set of indices for all users $\{1, \ldots, n\}$ and items $\{1, \ldots, m\}$.	Set
$\mathcal{M}$	The set of user-item pairs (i, j) for which a rating exists.	$\subseteq [n] \times [m]$
M	The total number of known ratings, $M = \lvert\mathcal{M}\rvert$.	$\mathbb{N}$
γ	The learning rate for stochastic gradient descent.	$\mathbb{R}^+$
λ	Regularization hyperparameter.	$\mathbb{R}^+$
$\omega_{\mathbf{U}}, \omega_{\mathbf{V}}$	Convergence thresholds for the sum of squared gradients of user and item profiles, respectively.	$\mathbb{R}^+$
α, β	Bit precision parameters for fixed-point representation. α is for ratings/profiles, β for the learning rate γ.	$\mathbb{N}$
	2. Cryptographic Elements	

Symbol	Description	Type / Domain
$\mathsf{pk}_{\mathsf{AHE}}, \mathsf{sk}_{\mathsf{AHE}}$	Public and secret keys for the Additive Homomorphic Encryption (AHE) scheme.	AHE Key Pair
$\mathsf{pk}_{\mathsf{HE}}, \mathsf{sk}_{\mathsf{HE}}$	Public and secret keys for the Fully Homomorphic Encryption (FHE) scheme.	FHE Key Pair
$enc_{\mathsf{pk}_{\mathsf{AHE}}}(\cdot), dec_{\mathsf{sk}_{\mathsf{AHE}}}(\cdot)$	Encryption and decryption functions for AHE scheme.	Function
$enc(\cdot), dec(\cdot)$	Shorthand for $enc_{\mathsf{pk}_{\mathsf{HE}}}(\cdot)$ and $enc_{\mathsf{sk}_{\mathsf{HE}}}(\cdot)$ denoting encryption and decryption functions for FHE scheme.	Function
	3. Standard Matrix Factorization Variables	
r_{ij}	The rating given by user i to item j.	$\mathbb{R}$
$\overline{r_{ij}}$	The fixed-point representation of the rating r_{ij}.	$\mathbb{Z}$
$\mathbf{u}_i, \mathbf{v}_j$	Latent feature vectors for user i and item j.	$\mathbb{R}^d$
$\overline{\mathbf{u}_i}, \overline{\mathbf{v}_j}$	Fixed-point representations of the latent vectors.	$\mathbb{Z}^d$
$\hat{r}_{ij}$	The predicted rating, computed as $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$.	$\mathbb{R}$
e_{ij}	The prediction error for rating (i,j), i.e., $\langle \mathbf{u}_i(k-1), \mathbf{v}_j(k-1) \rangle - r_{ij}$.	$\mathbb{Z}$
	4. Packed Data Vectors (Plaintext)	
$\mathbf{U}(k)$	Packed user profiles: concatenation of $\overline{\mathbf{u}_i(k)}$ for all $(i,j) \in \mathcal{M}$.	$\mathbb{Z}^{dM}$
$\mathbf{V}(k)$	Packed item profiles: concatenation of $\overline{\mathbf{v}_j(k)}$ for all $(i,j) \in \mathcal{M}$.	$\mathbb{Z}^{dM}$
$\hat{\mathbf{U}}(k)$	Packed "first occurrence" user profiles. Contains $\overline{\mathbf{u}_i(k)}$ only for the first appearance of user i in $\mathcal{M}$; $\mathbf{0}^d$ otherwise.	$\mathbb{Z}^{dM}$
$\hat{\mathbf{V}}(k)$	Packed "first occurrence" item profiles. Contains $\overline{\mathbf{v}_j(k)}$ only for the first appearance of item j in $\mathcal{M}$; $\mathbf{0}^d$ otherwise.	$\mathbb{Z}^{dM}$
$\mathbf{r}$	Packed rating vector where each d-dimensional block corresponding to $(i,j) \in \mathcal{M}$ is $(-\overline{r_{ij}}, 0, \ldots, 0)$.	$\mathbb{Z}^{dM}$
$\mathbf{E}(k-1)$	Packed error vector where each d-dimensional block corresponding to $(i,j) \in \mathcal{M}$ is filled with the error $e_{ij}(k-1)$.	$\mathbb{Z}^{dM}$
$\mathbf{E}'(k-1)$	Intermediate result in error calculation: $\mathbf{U}(k-1) \odot \mathbf{V}(k-1) + 2^{\alpha}\mathbf{r}$.	$\mathbb{Z}^{dM}$
$\nabla'_{\mathbf{U}}(k)$, $\nabla'_{\mathbf{V}}(k)$	Packed vectors representing the scaled gradients for all user/item profiles before aggregation.	$\mathbb{Z}^{dM}$
$\mathbf{U}'(k)$, $\mathbf{V}'(k)$	Packed vectors for the updated user/item profiles, before aggregation and final scaling.	$\mathbb{Z}^{dM}$
$\nabla\mathbf{U}_{\mathsf{agg}}(k)$, $\nabla\mathbf{V}_{\mathsf{agg}}(k)$	Aggregated gradient vectors, where all gradient contributions for a single user/item have been summed.	$\mathbb{Z}^{dn}, \mathbb{Z}^{dm}$
	5. Masking and Unmasking Variables	
$\sigma_{ij}, \boldsymbol{\sigma}(k-1), \boldsymbol{\sigma}_{\mathbf{U}}(k), \ldots$	General notation for random masks generated by RecSys to hide data before delegating computation to the CSP. The subscript indicates the variable being masked.	Varies ($\mathbb{Z}, \mathbb{Z}^{dM}, \ldots$)
$f_{\sigma}, f_{\sigma}(k-1), f_{\boldsymbol{\sigma}_{\mathbf{U}}}, \ldots$	General notation for unmasking terms computed by RecSys. An f_{σ} term is used to remove the effect of a corresponding σ mask after the CSP's computation.	Varies ($\mathbb{Z}^{dM}, \mathbb{Z}^{dn}, \ldots$)
x^*	General notation for a variable x in its masked state (e.g., $\mathbf{r}^*$). This is the form of data sent to the CSP for plaintext computation after being additively masked with a corresponding σ term.	(Domain of x)
	6. Convergence Check Variables	
$\boldsymbol{\nabla}^2_{\mathbf{U}}, \boldsymbol{\nabla}^2_{\mathbf{V}}$	Element-wise squares of the aggregated gradient vectors $\nabla\mathbf{U}_{\mathsf{agg}}(k)$ and $\nabla\mathbf{V}_{\mathsf{agg}}(k)$.	$\mathbb{Z}^{dn}, \mathbb{Z}^{dm}$

Symbol	Description	Type / Domain
$\boldsymbol{\sigma}_{\nabla^2_{\mathbf{U}}}, \boldsymbol{\sigma}_{\nabla^2_{\mathbf{V}}}$	Random vectors used to mask the squared gradients before sending to the CSP for the convergence check.	$\mathbb{Z}^{dn}, \mathbb{Z}^{dm}$
$\Sigma^{\sigma}_{\mathbf{U}}, \Sigma^{\sigma}_{\mathbf{V}}$	Sum of the elements of a mask vector ($\boldsymbol{\sigma}_{\nabla^2_{\mathbf{U}}}$ or $\boldsymbol{\sigma}_{\nabla^2_{\mathbf{V}}}$).	$\mathbb{Z}$
s_U, s_V	Plaintext scalars sent to the CSP. Combines a convergence threshold (ω) with the sum of a mask's elements (Σ^{σ}) for secure comparison.	$\mathbb{Z}$
s'_U, s'_V	Plaintext scalars computed by the CSP. Represents the sum of all elements in a masked squared gradient vector.	$\mathbb{Z}$
b_U, b_V	Boolean flags indicating whether the convergence criteria for user and item profiles have been met.	Boolean

4.2. System Model and Security Assumptions

The system consists of three types of parties: users, the recommender system (RecSys), and the Crypto-Service Provider (CSP). The description, goals and tasks of these parties are summarized in Table 4.2.

Table 4.2: Roles, goals and tasks of each party in the algorithm developed by Kim, Kim, Koo, *et al.* [24]

Party	Description	Goals	Tasks
User i	End-user who rates items and wishes to receive recommendations	Obtain accurate personalized recommendations while keeping the ratings private	Encrypt and upload ratings, receive and decrypt the results
Recommender System (RecSys)	Service that orchestrates the protocol	Produce recommendations for users	Receive the encrypted ratings, run matrix factorization with the CSP, and return encrypted predictions to users
Crypto-Service Provider (CSP)	A trusted third party that stores cryptographic keys	Protect secret keys and assist RecSys without learning raw data	Generate and manage cryptographic keys, and assist RecSys by performing specified computations on masked data

The adversarial model adopted in this protocol is honest-but-curious, meaning that all parties are assumed to follow the protocol steps accurately, but might try to infer additional sensitive information from the data they observe during the execution. The protocol's goal is then to ensure that no party can gain any valuable information from these observations.

Each party in the algorithm has a different privacy goal:

- **Users** wish to keep their ratings (r_{ij}) and derived profiles ($\mathbf{u}_i$) private from all other parties. Furthermore, item profiles ($\mathbf{v}_j$) must also be protected. An adversary with access to item profiles could infer a user's preferences for certain items, especially if they possess side-channel information about that user.
- **Recommender System** wishes to protect their tuning parameters from users and the CSP; these parameters are proprietary know-how, developed using significant resources, and must be protected from competitors. Furthermore, RecSys aims to protect the user and item profiles from unauthorized access, as their exposure could create significant attack vectors and compromise user privacy.

- **CSP** is assumed to be a public or governmental entity, therefore its main goal is to protect its reputation. To maintain its reputation, it needs to securely store the secret keys and perform the computations correctly on the masked data they receive.

The privacy assumption is that RecSys and CSP do not collude, meaning that they do not share any sensitive information among themselves. The user privacy could be breached if CSP shares the secret keys with RecSys, this would allow all user-submitted encrypted ratings to be decrypted into plaintext, breaching the privacy goal of the users. Similarly, if RecSys were to share the values used during the masking phase with the CSP, it would be easy to recover the original user ratings, again breaching the user privacy goals. The authors explain that this is a realistic assumption, since in practice, the CSP would be a public or governmental entity valuing its reputation.

4.2.1. Data Packing in Kim et al

The data structures developed for matrix factorization in Kim, Kim, Koo, *et al.* [24] are as follows.

First, we assume that all users and all items have some universally accepted ordering, for example, their index in increasing order.

The set $\mathcal{M} = \{(i, j) \mid \text{user } i \text{ rated item } j\}$ contains all the user-item index pairs for which there is a known rating. While $\mathcal{M}$ is a set, for the purpose of defining indices and in contexts requiring a specific sequence in this section, we will consider its elements to be arranged in a fixed canonical order: first by user index i, and then by item index j. Thus, where order is relevant, $\mathcal{M}$ should be understood as representing this specific ordered list of pairs. Let $M = |\mathcal{M}|$ be the total number of known ratings. In the following definitions, $\|$ denotes vector concatenation, and $\mathbf{0}^d$ denotes a d-dimensional zero vector.

Given $\mathcal{M}$, four different vectors are defined:

- **User Profiles:** $\mathbf{U} := \|_{(i,j)\in\mathcal{M}} \mathbf{u}_i$, where $\mathbf{u}_i$ is the d-dimensional profile vector for user i. The resulting vector has dimension dM.
- **Item Profiles:** $\mathbf{V} := \|_{(i,j)\in\mathcal{M}} \mathbf{v}_j$, where $\mathbf{v}_j$ is the d-dimensional profile vector for item j. The resulting vector has dimension dM.
- **Packed "First Occurrence" User Profiles:** $\hat{\mathbf{U}} := \|_{(i,j)\in\mathcal{M}} \hat{\mathbf{u}}_i$, where $\hat{\mathbf{u}}_i = \mathbf{u}_i$ if the user index i (from the pair $(i, j) \in \mathcal{M}$) is encountered for the first time in the canonical ordering of $\mathcal{M}$. Otherwise, $\hat{\mathbf{u}}_i = \mathbf{0}^d$. This structure is useful for adding values, which need to be applied only once for each unique user (more details are provided in later sections).
- **Packed "First Occurrence" Item Profiles:** $\hat{\mathbf{V}} := \|_{(i,j)\in\mathcal{M}} \hat{\mathbf{v}}_j$, where $\hat{\mathbf{v}}_j = \mathbf{v}_j$ if the item index j (from the pair $(i, j) \in \mathcal{M}$) is encountered for the first time in the canonical ordering of $\mathcal{M}$. Otherwise, $\hat{\mathbf{v}}_j = \mathbf{0}^d$. This structure is similarly useful for item-specific values.

The packed vector of ratings is defined as follows:

- **Ratings:** $\mathbf{r} = \|_{(i,j)\in\mathcal{M}} (-\overline{r}_{ij}, 0, \ldots, 0)$. Each rating r_{ij} is converted to fixed point representation $\overline{r_{ij}}$, negated, and padded with $d - 1$ zeros to form a d-dimensional block. These blocks are concatenated to obtain the final vector containing all ratings $\mathbf{r}$ with dimension dM.

This data packing scheme from Kim, Kim, Koo, *et al.* [24] defines specialized vector representations for user profiles, item profiles, and ratings used in the privacy-preserving matrix factorization algorithm. These elements allow for efficient data management in the cryptographic protocol, increasing its computational and memory efficiency.

4.3. Protocol Phases

The protocol proceeds in four distinct phases: Setup, Rating Upload, Learning, and Recommendation.

4.3.1. Setup Phase

During the setup phase, the keys and public parameters are distributed.

(1) CSP generates the keys $(\mathsf{pk}_{\mathsf{AHE}}, \mathsf{sk}_{\mathsf{AHE}})$ and $(\mathsf{pk}_{\mathsf{HE}}, \mathsf{sk}_{\mathsf{HE}})$. Securely stores the secret keys $\mathsf{sk}_{\mathsf{AHE}}, \mathsf{sk}_{\mathsf{HE}}$. Makes the public keys $\mathsf{pk}_{\mathsf{AHE}}, \mathsf{pk}_{\mathsf{HE}}$ available to RecSys and Users as needed.

(2) RecSys determines and publishes public parameters (d, n, m, α, β). Chooses its private tuning parameters (λ, γ) and convergence thresholds: $\omega_{\mathbf{U}}$ and $\omega_{\mathbf{V}}$. RecSys converts its private tuning parameters to fixed-point representation: $\overline{\lambda}$ with precision α, and $\overline{\gamma}$ with precision β.

(3) Users obtain the public key $\mathsf{pk}_{\mathsf{AHE}}$.

4.3.2. Rating Upload Phase

In this phase, users securely submit their ratings to RecSys. RecSys, in collaboration with the CSP, processes these submissions to obtain an FHE encryption of the packed rating vector, $enc(\mathbf{r})$. Additive Homomorphic Encryption (AHE) is used for the initial rating submission and masking, after which the data is converted to an FHE-encrypted format suitable for the Learning Phase.

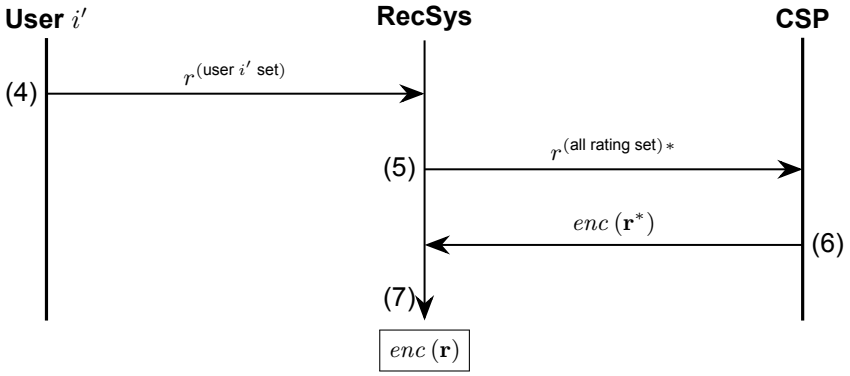

Figure 4.1: The communication flow for the Rating Upload Phase.

Explanation:

(4) Each user i' first converts their ratings $r_{i'j}$ into a fixed point representation $\overline{r_{i'j}}$ for all items j they have rated (i.e., for which $(i', j) \in \mathcal{M}$). Then these ratings are encrypted using $\mathsf{pk}_{\mathsf{AHE}}$ and placed in a set of triplets, each containing the user ID (i'), the item ID (j), and the corresponding encrypted rating ($enc_{\mathsf{pk}_{\mathsf{AHE}}}(\overline{r_{i'j}})$). This set, formally

$$r^{(\text{user } i' \text{ set})} := \{(i', j, enc_{\mathsf{pk}_{\mathsf{AHE}}}(\overline{r_{i'j}})) : (i', j) \in \mathcal{M}\}, \tag{4.1}$$

is then sent to RecSys.

(5) RecSys collects all encrypted user ratings. For each user-item pair $(i, j) \in \mathcal{M}$, they generate a random mask $\sigma_{ij} \in \mathbb{Z}$. This mask is encrypted with $\mathsf{pk}_{\mathsf{AHE}}$, and added to each encrypted rating $enc_{\mathsf{pk}_{\mathsf{AHE}}}(\overline{r_{ij}})$ using the additive property of AHE. The updated collections of triplets are then joined into one set, formally

$$r^{(\text{all rating set})*} := \{(i, j, enc_{\mathsf{pk}_{\mathsf{AHE}}}(\overline{r_{ij}} + \sigma_{ij})) : (i, j) \in \mathcal{M}\}, \tag{4.2}$$

which is then forwarded to the CSP.

(6) CSP receives $r^{(\text{all rating set})*}$. They decrypt all masked ratings using the secret key $\mathsf{sk}_{\mathsf{AHE}}$, obtaining a set of triplets $\{(i, j, \overline{r_{ij}} + \sigma_{ij}) : (i, j) \in \mathcal{M}\}$.

The masked ratings are sorted, negated, and combined to obtain a packed dM-dimensional vector:

$$\mathbf{r}^* := (\|_{(i,j)\in\mathcal{M})}(\overbrace{-\overline{r_{ij}} - \sigma_{ij}, 0, \ldots, 0}^{d})). \tag{4.3}$$

This vector, containing all user ratings, is in the format that is required for the future protocol; however, the user ratings are still masked. The array is encrypted using the public key $\mathsf{pk}_{\mathsf{HE}}$, and the encryption $enc(\mathbf{r}^*)$ is sent to RecSys, which will unmask the ratings, obtaining the encryption of $\mathbf{r}$ in the next step.

(7) RecSys must now remove the effect of its random masks σ_{ij} from the encrypted vector. The influence can be computed as:

$$f_\sigma := (\|_{(i,j\in\mathcal{M})}(\overbrace{-\sigma_{ij}, 0, \ldots, 0}^{d})). \tag{4.4}$$

To obtain the final packed rating vector $\mathbf{r}$, f_σ is negated, encrypted, and added to the received array using the additive property of HE.

$$enc\,(\mathbf{r}^*) \oplus enc\,(-f_\sigma) \tag{4.5}$$

$$= enc\left((\|_{(i,j\in\mathcal{M})}(\overbrace{-\overline{r_{ij}} - \sigma_{ij}, 0, \ldots, 0}^{d})) - (\|_{(i,j\in\mathcal{M})}(\overbrace{-\sigma_{ij}, 0, \ldots, 0}^{d}))\right) \tag{4.6}$$

$$= enc\left((\|_{(i,j\in\mathcal{M})}(\overbrace{-\overline{r_{ij}}, 0, \ldots, 0}\Big)^{d})\right) \tag{4.7}$$

$$= enc\,(\mathbf{r}) \tag{4.8}$$

Thus, the Rating Upload phase concludes with RecSys obtaining the FHE-encrypted packed rating vector, $enc\,(\mathbf{r})$. Using AHE-based encryption and a collaborative masking protocol with the CSP, all submitted ratings are now securely aggregated and formatted within this single ciphertext, ready for the privacy-preserving computations of the Learning Phase.

4.3.3. Learning Phase

The goal of the Learning Phase is to construct the user and item profiles $\mathbf{u}_i$ and $\mathbf{v}_j$ for each user $i \in [n]$ and item $j \in [m]$. This is achieved by starting with random profiles and iteratively refining them using stochastic gradient descent. This phase begins with RecSys initializing, packing, and encrypting these profiles in Step (8). The values of these profiles remain hidden (either encrypted or masked) throughout the remainder of the Learning Phase.

After the packed vector initialization in Step (8), RecSys, in collaboration with the CSP, performs gradient descent computations using homomorphic encryption and data masking. This is an iterative process with steps (9)-(20) being repeated until the stopping criterion is met (see step (20)).

To describe the changes in the packed data vectors after each iteration, we use $\mathbf{U}(k)$, $\mathbf{V}(k)$, $\hat{\mathbf{U}}(k)$, and $\hat{\mathbf{V}}(k)$ representing their corresponding vectors $\mathbf{U}$, $\mathbf{V}$, $\hat{\mathbf{U}}$, and $\hat{\mathbf{V}}$ after the k-th round, where $k = 0$ denotes the initial vectors generated by RecSys. Furthermore, during the Learning Phase only the FHE keys are being used: $\mathsf{pk}_{\mathsf{HE}}$ for encryption and $\mathsf{sk}_{\mathsf{HE}}$ for decryption. To maintain simplicity, we will denote encryption as $enc\,(\cdot)$ and decryption as $dec\,(\cdot)$, rather than their more explicit forms $enc_{\mathsf{pk}_{\mathsf{HE}}}(\cdot)$ and $dec_{\mathsf{sk}_{\mathsf{HE}}}(\cdot)$.

(8) RecSys generates the initial profile vectors $\mathbf{u}_i(0) \in \mathbb{R}^d$ for each user $i \in [n]$ and $\mathbf{v}_j(0) \in \mathbb{R}^d$ for each item $j \in [m]$. These are sampled as random d-dimensional ℓ_2-norm vectors. RecSys converts these vectors to fixed point representation $\mathbf{u}_i(0)$ and $\mathbf{v}_j(0)$ accordingly, and constructs the four initial packed vectors:

- User profiles: $\mathbf{U}(0) := \|_{(i,j)\in\mathcal{M}}\mathbf{u}_i(0)$,
- Item profiles: $\mathbf{V}(0) := \|_{(i,j)\in\mathcal{M}}\mathbf{v}_j(0)$.
- Packed "First Occurrence" User Profiles: $\hat{\mathbf{U}}(0) := \|_{(i,j)\in\mathcal{M}}\hat{\mathbf{u}}_i(0)$, where $\hat{\mathbf{u}}_i(0) = \mathbf{u}_i(0)$ if the user index i (from the pair $(i,j) \in \mathcal{M}$) is encountered for the first time in the canonical ordering of $\mathcal{M}$. Otherwise, $\hat{\mathbf{u}}_i(0) = \mathbf{0}^d$.
- Packed "First Occurrence" Item Profiles: $\hat{\mathbf{V}}(0) := \|_{(i,j)\in\mathcal{M}}\hat{\mathbf{v}}_j(0)$, where $\hat{\mathbf{v}}_j(0) = \mathbf{v}_j(0)$ if the item index j (from the pair $(i,j) \in \mathcal{M}$) is encountered for the first time in the canonical ordering of $\mathcal{M}$. Otherwise, $\hat{\mathbf{v}}_j(0) = \mathbf{0}^d$.

RecSys then encrypts all vectors, obtaining $enc\,(\mathbf{U}(0))$, $enc\,(\mathbf{V}(0))$, $enc\left(\hat{\mathbf{U}}(0)\right)$, and $enc\left(\hat{\mathbf{V}}(0)\right)$.

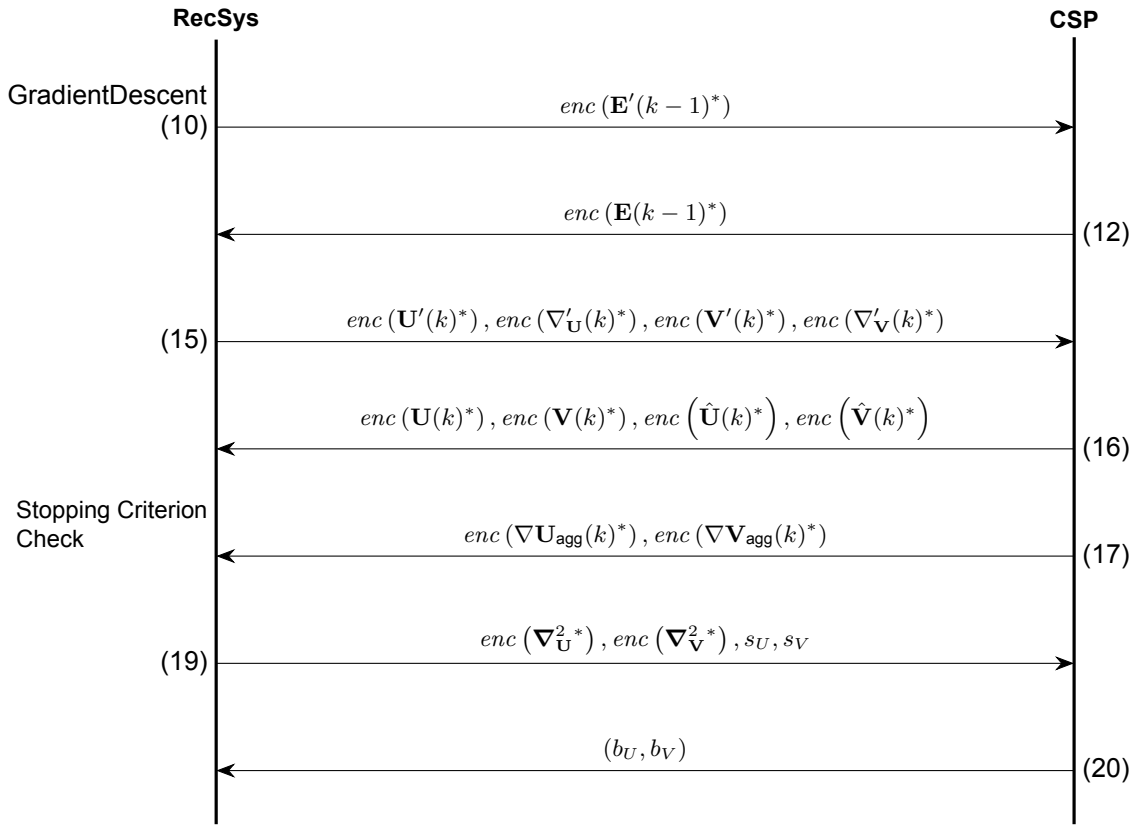

Figure 4.2: The communication flow for a single epoch of the Learning Phase.

The subsequent steps, from (9) to (20), define a single round (epoch) of the iterative learning process. Let k denote the current epoch number, starting from $k = 1$.

At the beginning of each round, RecSys holds the following FHE-encrypted data:

- The packed ratings: $enc\,(\mathbf{r})$ (obtained from the Rating Upload Phase),
- The packed user profiles: $enc\,(\mathbf{U}(k-1))$,
- The packed item profiles: $enc\,(\mathbf{V}(k-1))$.

(9) RecSys computes the encryption of:

$$\mathbf{E}'(k-1) := \mathbf{U}(k-1) \odot \mathbf{V}(k-1) + 2^{\alpha}\mathbf{r}. \tag{4.9}$$

This is achieved homomorphically on the ciphertexts from epoch $k-1$:

$$enc\,(\mathbf{E}'(k-1)) = (enc\,(\mathbf{U}(k-1)) \otimes enc\,(\mathbf{V}(k-1))) \oplus (2^{\alpha} \otimes enc\,(\mathbf{r}))\,. \tag{4.10}$$

This operation effectively computes the scaled prediction errors. The inputs $\mathbf{U}(k-1)$ and $\mathbf{V}(k-1)$ are in fixed-point representation, and the subsequent division by 2^{α} in Step (11) corrects for the scaling introduced by multiplying two fixed-point numbers.

(10) RecSys initiates the masking operation by sampling M random d-dimensional vectors $\boldsymbol{\sigma}(k-1)_{ij} \in \mathbb{Z}^d$, resulting in the following set:

$$\{\boldsymbol{\sigma}(k-1)_{ij} : (i,j) \in \mathcal{M}\}. \tag{4.11}$$

These vectors are concatenated to obtain a dM-dimensional vector mask $\boldsymbol{\sigma}(k-1)$.

$$\boldsymbol{\sigma}(k-1) := \|_{(i,j)\in\mathcal{M}}\boldsymbol{\sigma}(k-1)_{ij}. \tag{4.12}$$

Next, RecSys computes the encryption of the masked vector:

$$\mathbf{E}'(k-1)^* := \mathbf{E}'(k-1) + \boldsymbol{\sigma}(k-1). \tag{4.13}$$

This can be achieved using homomorphic addition:

$$enc\left(\mathbf{E}'(k-1)^*\right) = enc\left(\mathbf{E}'(k-1)\right) \oplus enc\left(\boldsymbol{\sigma}(k-1)\right). \tag{4.14}$$

This ciphertext is then sent to the CSP.

(11) The CSP decrypts the received ciphertext obtaining $\mathbf{E}'(k-1)^*$ and applies integer division by 2^α. This division serves as the necessary rescaling step in fixed-point arithmetic after the element-wise multiplication in Step (9). This results in:

$$\left\lfloor \frac{\mathbf{E}'(k-1)^*}{2^\alpha} \right\rfloor = \overline{\mathbf{U}(k-1) \odot \mathbf{V}(k-1)} + \mathbf{r} + \left\lfloor \frac{\boldsymbol{\sigma}(k-1)}{2^\alpha} \right\rfloor \tag{4.15}$$

$$= \|_{(i,j)\in\mathcal{M}} \left(\overline{\mathbf{u}_i(k-1) \odot \mathbf{v}_j(k-1)} - (\overline{r_{ij}}, 0, \ldots, 0) + \left\lfloor \frac{\boldsymbol{\sigma}(k-1)_{ij}}{2^\alpha} \right\rfloor \right). \tag{4.16}$$

(12) Using the vector from the previous step, CSP computes a masked error term ${e_{ij}}^*$ for each $(i,j) \in \mathcal{M}$ by summing all d elements of the vector:

$${e_{ij}}^* := \mathrm{sum}\left(\overline{\mathbf{u}_i(k-1) \odot \mathbf{v}_j(k-1)} - (\overline{r_{ij}}, 0, \ldots, 0) + \left\lfloor \frac{\boldsymbol{\sigma}(k-1)_{ij}}{2^\alpha} \right\rfloor \right). \tag{4.17}$$

This resulting scalar ${e_{ij}}^*$ can be understood as a masked prediction error for a rating of item j given by the user i. It combines the predicted rating (dot product between $\mathbf{u}_i(k-1)$ and $\mathbf{v}_j(k-1)$) minus the observed rating r_{ij}, plus some masking term.

Then CSP packs these values into a dM-dimensional vector $\mathbf{E}(k-1)^*$ as follows:

$$\mathbf{E}(k-1)^* := \left(\|_{(i,j\in\mathcal{M})} (\overbrace{{e_{ij}}^*, {e_{ij}}^*, \ldots, {e_{ij}}^*}^{d}) \right). \tag{4.18}$$

Finally, $\mathbf{E}(k-1)^*$ is encrypted and the ciphertext $enc\left(\mathbf{E}(k-1)^*\right)$ is sent to RecSys.

(13) Knowing the previously generated masking terms, RecSys computes the unmasking term:

$$f_\sigma(k-1) := \left(\|_{(i,j\in\mathcal{M})} (\overbrace{f_\sigma^{ij}, f_\sigma^{ij}, \ldots, f_\sigma^{ij}}^{d}) \right), \tag{4.19}$$

where f_σ^{ij} is defined as:

$$f_\sigma^{ij} := \mathrm{sum}\left(\left\lfloor \frac{\boldsymbol{\sigma}(k-1)_{ij}}{2^\alpha} \right\rfloor \right), \quad \forall (i,j) \in \mathcal{M}. \tag{4.20}$$

Next, RecSys removes the mask from $enc\left(\mathbf{E}(k-1)^*\right)$ to obtain the encryption of the error vector:

$$\mathbf{E}(k-1) := \left(\|_{(i,j\in\mathcal{M})} (\overbrace{e_{ij}, e_{ij}, \ldots, e_{ij}}^{d}) \right), \tag{4.21}$$

where e_{ij} is an error of the rating prediction. Namely,

$$e_{ij} = \overline{\langle \mathbf{u}_i(k-1), \mathbf{v}_j(k-1) \rangle - r_{ij}}. \tag{4.22}$$

This is achieved by encrypting the negation of $f_\sigma(k-1)$ and adding it to $enc\left(\mathbf{E}(k-1)^*\right)$ using homomorphic addition:

$$enc\left(\mathbf{E}(k-1)\right) = enc\left(\mathbf{E}(k-1)^*\right) \oplus enc\left(-f_\sigma(k-1)\right). \tag{4.23}$$

(14) RecSys computes encryptions of packed gradient vectors:

$$\nabla'_{\mathbf{U}}(k) = \mathbf{V}(k-1) \odot \mathbf{E}(k-1) + \overline{\lambda}\hat{\mathbf{U}}(k-1) \tag{4.24}$$

and

$$\nabla'_{\mathbf{V}}(k) = \mathbf{U}(k-1) \odot \mathbf{E}(k-1) + \overline{\lambda}\hat{\mathbf{V}}(k-1). \tag{4.25}$$

This can be achieved using homomorphic encryption:

$$enc\left(\nabla'_{\mathbf{U}}(k)\right) = \left(enc\left(\mathbf{V}(k-1)\right) \otimes enc\left(\mathbf{E}(k-1)\right)\right) \oplus \left(\overline{\lambda} \otimes enc\left(\hat{\mathbf{U}}(k-1)\right)\right) \tag{4.26}$$

and

$$enc\left(\nabla'_{\mathbf{V}}(k)\right) = \left(enc\left(\mathbf{U}(k-1)\right) \otimes enc\left(\mathbf{E}(k-1)\right)\right) \oplus \left(\overline{\lambda} \otimes enc\left(\hat{\mathbf{V}}(k-1)\right)\right). \tag{4.27}$$

Next, RecSys computes

$$\mathbf{U}'(k) := 2^{\alpha+\beta}\hat{\mathbf{U}}(k-1) - \overline{\gamma}\nabla'_{\mathbf{U}}(k) \tag{4.28}$$

and

$$\mathbf{V}'(k) := 2^{\alpha+\beta}\hat{\mathbf{V}}(k-1) - \overline{\gamma}\nabla'_{\mathbf{V}}(k). \tag{4.29}$$

This can be achieved using homomorphic encryption:

$$enc\left(\mathbf{U}'(k)\right) = \left(2^{\alpha+\beta} \otimes enc\left(\hat{\mathbf{U}}(k-1)\right)\right) \oplus \left(-\overline{\gamma} \otimes enc\left(\nabla'_{\mathbf{U}}(k)\right)\right) \tag{4.30}$$

and

$$enc\left(\mathbf{V}'(k)\right) = \left(2^{\alpha+\beta} \otimes enc\left(\hat{\mathbf{V}}(k-1)\right)\right) \oplus \left(-\overline{\gamma} \otimes enc\left(\nabla'_{\mathbf{V}}(k)\right)\right). \tag{4.31}$$

These vectors are now one transformation away from obtaining the updated user and item profile vectors. Recall that an update step in matrix factorization is described as:

$$\mathbf{u}_i(k) \leftarrow \mathbf{u}_i(k-1) - \gamma\nabla_{\mathbf{U}}(k) \tag{4.32}$$

and

$$\mathbf{v}_j(k) \leftarrow \mathbf{v}_j(k-1) - \gamma\nabla_{\mathbf{V}}(k). \tag{4.33}$$

where

$$\nabla_{\mathbf{U}}(k) = \mathbf{v}_j(k-1)\left(r_{ij} - \langle\mathbf{u}_i(k-1), \mathbf{v}_j(k-1)\rangle\right) + \lambda\mathbf{u}_i(k-1), \tag{4.34}$$

$$\nabla_{\mathbf{V}}(k) = \mathbf{u}_i(k-1)\left(r_{ij} - \langle\mathbf{u}_i(k-1), \mathbf{v}_j(k-1)\rangle\right) + \lambda\mathbf{v}_j(k-1). \tag{4.35}$$

Essentially this step performs these computations, but the outcomes are packed inside the vector that needs gathering and scattering to obtain the final values. This will be done in the next two steps.

(15) RecSys initiates data masking by generating four sets of M random d-dimensional masking vectors:

$$\{\boldsymbol{\sigma}_{\mathbf{U}}(k)_{ij} : (i,j) \in \mathcal{M}\}, \qquad \{\boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k)_{ij} : (i,j) \in \mathcal{M}\}, \tag{4.36}$$

$$\{\boldsymbol{\sigma}_{\mathbf{V}}(k)_{ij} : (i,j) \in \mathcal{M}\}, \qquad \{\boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k)_{ij} : (i,j) \in \mathcal{M}\}. \tag{4.37}$$

These vectors are then concatenated to obtain four dM-dimensional vector masks:

$$\boldsymbol{\sigma}_{\mathbf{U}}(k) := \|_{(i,j)\in\mathcal{M}}\boldsymbol{\sigma}_{\mathbf{U}}(k)_{ij}, \qquad \boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k) := \|_{(i,j)\in\mathcal{M}}\boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k)_{ij}, \tag{4.38}$$

$$\boldsymbol{\sigma}_{\mathbf{V}}(k) := \|_{(i,j)\in\mathcal{M}}\boldsymbol{\sigma}_{\mathbf{V}}(k)_{ij}, \qquad \boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k) := \|_{(i,j)\in\mathcal{M}}\boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k)_{ij}. \tag{4.39}$$

RecSys then computes the encryptions of the following masked vectors:

$$\mathbf{U}'(k)^* := \mathbf{U}'(k) + \boldsymbol{\sigma}_{\mathbf{U}}(k), \qquad \nabla'_{\mathbf{U}}(k)^* := \nabla'_{\mathbf{U}}(k) + \boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k), \tag{4.40}$$

$$\mathbf{V}'(k)^* := \mathbf{V}'(k) + \boldsymbol{\sigma}_{\mathbf{V}}(k), \qquad \nabla'_{\mathbf{V}}(k)^* := \nabla'_{\mathbf{V}}(k) + \boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k). \tag{4.41}$$

This is achieved using homomorphic addition:

$$enc\left(\mathbf{U}'(k)^*\right) = enc\left(\mathbf{U}'(k)\right) \oplus enc\left(\boldsymbol{\sigma}_{\mathbf{U}}(k)\right), \tag{4.42}$$

$$enc\left(\nabla'_{\mathbf{U}}(k)^*\right) = enc\left(\nabla'_{\mathbf{U}}(k)\right) \oplus enc\left(\boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k)\right), \tag{4.43}$$

$$enc\left(\mathbf{V}'(k)^*\right) = enc\left(\mathbf{V}'(k)\right) \oplus enc\left(\boldsymbol{\sigma}_{\mathbf{V}}(k)\right), \tag{4.44}$$

$$enc\left(\nabla'_{\mathbf{V}}(k)^*\right) = enc\left(\nabla'_{\mathbf{V}}(k)\right) \oplus enc\left(\boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k)\right). \tag{4.45}$$

The encryption of these masked values is then sent to the CSP.

(16) CSP decrypts the received ciphertexts, obtaining four dM-dimensional vectors $\mathbf{U}'(k)^*$, $\nabla'_{\mathbf{U}}(k)^*$, $\mathbf{V}'(k)^*$, and $\nabla'_{\mathbf{V}}(k)^*$.

These dM-dimensional vectors can be seen as concatenations of M distinct d-dimensional blocks, where each block corresponds to a unique user-item pair (i,j) from $\mathcal{M}$ according to its position in the canonical ordering.

Therefore, the packed user profile vector $\mathbf{U}'(k)^*$ can be expressed as:

$$\mathbf{U}'(k)^* = \|_{(i,j)\in\mathcal{M}}\ \mathbf{x}_{ij}(k)^*, \tag{4.46}$$

where $\mathbf{x}_{ij}(k)^*$ is the d-dimensional block from $\mathbf{U}'(k)^*$ whose index corresponds to the location of the tuple (i,j) in $\mathcal{M}$. Note that this is possible because there are exactly M unique tuples in $\mathcal{M}$.

Essentially, each block $\mathbf{x}_{ij}(k)^*$ contains an update component for user i derived from their interaction with item j. The specific structure of the input vectors, prepared by RecSys in Step (14), ensures that when these components are aggregated, the previous profile values and regularization terms are correctly applied exactly once per user.

Analogously, we can express the packed item profiles as:

$$\mathbf{V}'(k)^* = \|_{(i,j)\in\mathcal{M}}\ \mathbf{y}_{ij}(k)^* \tag{4.47}$$

with d-dimensional vectors $\mathbf{y}_{ij}(k) \forall (i,j) \in \mathcal{M}$.

To obtain the updated (and still masked) profile for each user and item, CSP first aggregates these components, obtaining $\mathbf{u}_i^\dagger(k)^*$ and $\mathbf{v}_j^\dagger(k)^*$ for all users i and items j:

- For each user $i' \in [n]$, its $\mathbf{u}_{i'}^\dagger(k)^*$ is obtained by summing all blocks related to this user:

$$\mathbf{u}_{i'}^\dagger(k)^* := \sum_{j:(i',j)\in\mathcal{M}} \mathbf{x}_{i'j}(k)^*. \tag{4.48}$$

- Similarly, for each item $j' \in [m]$, its $\mathbf{v}_{j'}^\dagger(k)^*$ is obtained as follows:

$$\mathbf{v}_{j'}^\dagger(k)^* := \sum_{i:(i,j')\in\mathcal{M}} \mathbf{y}_{ij'}(k)^*. \tag{4.49}$$

Next, these vectors are integer-divided by $2^{\alpha+\beta}$ to cancel the effects caused by fixed-point multiplication in step (14). Namely,

$$\mathbf{u}_i(k)^* := \left\lfloor \frac{\mathbf{u}_i^\dagger(k)^*}{2^{\alpha+\beta}} \right\rfloor \quad \forall i \in [n], \qquad \mathbf{v}_j(k)^* := \left\lfloor \frac{\mathbf{v}_j^\dagger(k)^*}{2^{\alpha+\beta}} \right\rfloor \quad \forall j \in [m]. \tag{4.50}$$

With these newly-updated user and item profiles, the CSP then constructs the updated packed profile vectors $\mathbf{U}(k)^*$,$\mathbf{V}(k)^*$, $\hat{\mathbf{U}}(k)^*$, and $\hat{\mathbf{V}}(k)^*$. Note that these vectors are still masked. The user and profile vectors are obtained as follows:

$$\mathbf{U}(k)^* := \|_{(i,j)\in\mathcal{M}} \mathbf{u}_i(k)^*, \qquad \mathbf{V}(k)^* := \|_{(i,j)\in\mathcal{M}} \mathbf{v}_j(k)^*. \tag{4.51}$$

Next, the "First Occurrence" User and Item Profiles are constructed as follows:

$$\hat{\mathbf{U}}(k)^* := \|_{(i,j)\in\mathcal{M}} \hat{\mathbf{u}}_{ij}(k)^*, \qquad \hat{\mathbf{V}}(k)^* := \|_{(i,j)\in\mathcal{M}} \hat{\mathbf{v}}_{ij}(k)^*, \tag{4.52}$$

where

$$\hat{\mathbf{u}}_{ij}(k)^* := \begin{cases} \mathbf{u}_i(k)^* & \text{if } (i,j) \text{ is the first pair in the canonical ordering of } \mathcal{M} \text{ that contains user } i, \\ \mathbf{0}^d & \text{otherwise,} \end{cases} \tag{4.53}$$

$$\hat{\mathbf{v}}_{ij}(k)^* := \begin{cases} \mathbf{v}_j(k)^* & \text{if } (i,j) \text{ is the first pair in the canonical ordering of } \mathcal{M} \text{ that contains item } j, \\ \mathbf{0}^d & \text{otherwise.} \end{cases} \tag{4.54}$$

The encryptions of the resulting vectors $enc\left(\mathbf{U}(k)^*\right)$, $enc\left(\mathbf{V}(k)^*\right)$, $enc\left(\hat{\mathbf{U}}(k)^*\right)$ and $enc\left(\hat{\mathbf{V}}(k)^*\right)$ are then sent to RecSys.

(17) Alongside the profile update vectors detailed in step (16), the CSP processes the masked scaled gradient vectors, $\nabla'_{\mathbf{U}}(k)^*$ and $\nabla'_{\mathbf{V}}(k)^*$, received from RecSys in step (15). After decrypting these two dM-dimensional vectors, the CSP aggregates and scales these gradient components.

Similarly to expressing $\mathbf{V}'(k)$ and $\mathbf{U}'(k)$ as a concatenation of M d-dimensional vectors in step (16), we can express the packed gradient vectors as:

$$\nabla'_{\mathbf{U}}(k)^* = \|_{(i,j)\in\mathcal{M}}\ \mathbf{g}_{ij}^{\mathbf{U}}(k)^*, \qquad \nabla'_{\mathbf{V}}(k)^* = \|_{(i,j)\in\mathcal{M}}\ \mathbf{g}_{ij}^{\mathbf{V}}(k)^*, \tag{4.55}$$

where each $\mathbf{g}_{ij}^{\mathbf{U}}(k)^*$ and $\mathbf{g}_{ij}^{\mathbf{V}}(k)^*$ is the d-dimensional block from $\nabla'_{\mathbf{U}}(k)^*$ and $\nabla'_{\mathbf{V}}(k)^*$ whose index corresponds to the location of the tuple (i,j) in $\mathcal{M}$.

To obtain an aggregated masked gradient for each user, the CSP performs the following. For each user $i' \in [n]$, it sums all d-dimensional blocks $\mathbf{g}_{i'j}^{\mathbf{U}}(k)^*$ related to this user:

$$\nabla\mathbf{u}_{i'}^\dagger(k)^* := \sum_{j:(i',j)\in\mathcal{M}} \mathbf{g}_{i'j}^{\mathbf{U}}(k)^*. \tag{4.56}$$

This sum $\nabla\mathbf{u}_{i'}^\dagger(k)^*$ represents the total 2^α-scaled masked gradient for user i. To correct the scaling, an integer division by 2^α is applied:

$$\nabla\mathbf{u}_{i'}(k)^* := \left\lfloor \frac{\nabla\mathbf{u}_{i'}^\dagger(k)^*}{2^\alpha} \right\rfloor. \tag{4.57}$$

A similar process is followed for item gradients. For each item $j' \in [m]$, the CSP sums the relevant blocks $\mathbf{g}_{ij'}^{\mathbf{V}}(k)^*$:

$$\nabla\mathbf{v}_{j'}^\dagger(k)^* := \sum_{i:(i,j')\in\mathcal{M}} \mathbf{g}_{ij'}^{\mathbf{V}}(k)^*, \tag{4.58}$$

and then scales the result:

$$\nabla \mathbf{v}_{j'}(k)^* := \left\lfloor \frac{\nabla \mathbf{v}^{\dagger}_{j'}(k)^*}{2^{\alpha}} \right\rfloor. \tag{4.59}$$

The CSP then concatenates all these aggregated vectors to obtain two packed vectors: $\nabla \mathbf{U}_{\mathsf{agg}}(k)^*$, which contains all user gradients, and $\nabla \mathbf{V}_{\mathsf{agg}}(k)^*$, containing all item gradients.

$$\nabla \mathbf{U}_{\mathsf{agg}}(k)^* := \|_{i\in[n]} \nabla \mathbf{u}_i(k)^*, \tag{4.60}$$

$$\nabla \mathbf{V}_{\mathsf{agg}}(k)^* := \|_{j\in[m]} \nabla \mathbf{v}_j(k)^*. \tag{4.61}$$

Finally, the CSP encrypts these two aggregated, scaled, and still-masked gradient vectors, obtaining $enc\left(\nabla \mathbf{U}_{\mathsf{agg}}(k)^*\right)$ and $enc\left(\nabla \mathbf{V}_{\mathsf{agg}}(k)^*\right)$. These ciphertexts are sent to RecSys. RecSys will subsequently unmask these vectors and use them for checking the convergence of the learning algorithm.

(18) In this step, RecSys unmasks the encryptions of the received vectors: $enc\left(\mathbf{U}(k)^*\right)$, $enc\left(\mathbf{V}(k)^*\right)$, $enc\left(\hat{\mathbf{U}}(k)^*\right)$, $enc\left(\hat{\mathbf{V}}(k)^*\right)$, $enc\left(\nabla \mathbf{U}_{\mathsf{agg}}(k)^*\right)$, and $enc\left(\nabla \mathbf{V}_{\mathsf{agg}}(k)^*\right)$. The unmasking terms are computed based on the random masks $\boldsymbol{\sigma}_{\mathbf{U}}(k)$, $\boldsymbol{\sigma}_{\mathbf{V}}(k)$, $\boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k)$, and $\boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k)$ that RecSys generated in Step (15). Recall that each of these dM-dimensional masks is a concatenation of M d-dimensional blocks; for instance, $\boldsymbol{\sigma}_{\mathbf{U}}(k) = \|_{(i,j)\in\mathcal{M}} \boldsymbol{\sigma}_{\mathbf{U}}(k)_{ij}$.

First, RecSys constructs intermediate scaled aggregated mask vectors. These vectors capture the effect of gathering and scaling by 2^{α} performed by the CSP:

$$(\Delta\boldsymbol{\sigma}_{\mathbf{U}}(k))_i := \left\lfloor \frac{1}{2^{\alpha+\beta}} \left(\sum_{j':(i,j')\in\mathcal{M}} \boldsymbol{\sigma}_{\mathbf{U}}(k)_{ij'} \right) \right\rfloor \quad \forall i \in [n], \tag{4.62}$$

$$(\Delta\boldsymbol{\sigma}_{\mathbf{V}}(k))_j := \left\lfloor \frac{1}{2^{\alpha+\beta}} \left(\sum_{i':(i',j)\in\mathcal{M}} \boldsymbol{\sigma}_{\mathbf{V}}(k)_{i'j} \right) \right\rfloor \quad \forall j \in [m], \tag{4.63}$$

$$(\Delta\boldsymbol{\sigma}_{\nabla\mathbf{U}}(k))_i := \left\lfloor \frac{1}{2^{\alpha}} \left(\sum_{j':(i,j')\in\mathcal{M}} \boldsymbol{\sigma}_{\nabla_{\mathbf{U}}}(k)_{ij'} \right) \right\rfloor \quad \forall i \in [n], \tag{4.64}$$

$$(\Delta\boldsymbol{\sigma}_{\nabla\mathbf{V}}(k))_j := \left\lfloor \frac{1}{2^{\alpha}} \left(\sum_{i':(i',j)\in\mathcal{M}} \boldsymbol{\sigma}_{\nabla_{\mathbf{V}}}(k)_{i'j} \right) \right\rfloor \quad \forall j \in [m]. \tag{4.65}$$

Using these intermediate scaled aggregated mask vectors, RecSys constructs the six final unmasking terms:

$$f_{\boldsymbol{\sigma}_{\mathbf{U}}}(k) := \|_{(i,j)\in\mathcal{M}} (\Delta\boldsymbol{\sigma}_{\mathbf{U}}(k))_i, \qquad f_{\boldsymbol{\sigma}_{\mathbf{V}}}(k) := \|_{(i,j)\in\mathcal{M}} (\Delta\boldsymbol{\sigma}_{\mathbf{V}}(k))_j, \tag{4.66}$$

$$f_{\boldsymbol{\sigma}_{\nabla\mathbf{U}}}(k) := \|_{i\in[n]} (\Delta\boldsymbol{\sigma}_{\nabla\mathbf{U}}(k))_i, \qquad f_{\boldsymbol{\sigma}_{\nabla\mathbf{V}}}(k) := \|_{j\in[m]} (\Delta\boldsymbol{\sigma}_{\nabla\mathbf{V}}(k))_j, \tag{4.67}$$

$$f_{\boldsymbol{\sigma}_{\hat{\mathbf{U}}}}(k) := \|_{(i,j)\in\mathcal{M}} \mathbf{b}_{\hat{\mathbf{U}},ij}(k), \qquad f_{\boldsymbol{\sigma}_{\hat{\mathbf{V}}}}(k) := \|_{(i,j)\in\mathcal{M}} \mathbf{b}_{\hat{\mathbf{V}},ij}(k), \tag{4.68}$$

where

$$\mathbf{b}_{\hat{\mathbf{U}},ij}(k) := \begin{cases} (\Delta\boldsymbol{\sigma}_{\mathbf{U}}(k))_i & \text{if } (i,j) \text{ is the first pair in the canonical ordering of } \mathcal{M} \text{ that contains user } i, \\ \mathbf{0}^d & \text{otherwise,} \end{cases} \tag{4.69}$$

and

$$\mathbf{b}_{\hat{\mathbf{V}},ij}(k) := \begin{cases} (\Delta\boldsymbol{\sigma}_{\mathbf{V}}(k))_j & \text{if } (i,j) \text{ is the first pair in the canonical ordering of } \mathcal{M} \text{ that contains item } j, \\ \mathbf{0}^d & \text{otherwise.} \end{cases} \tag{4.70}$$

RecSys then encrypts the negation of each of these unmasking terms and uses the additive homomorphic property of the FHE scheme to subtract them. This step removes the influence of the masks, up to the small precision errors introduced by the 'floor' operations, yielding encryptions of the updated profile vectors:

$$enc\left(\mathbf{U}(k)\right) = enc\left(\mathbf{U}(k)^*\right) \oplus enc\left(-f_{\boldsymbol{\sigma}_\mathbf{U}}(k)\right), \tag{4.71}$$
$$enc\left(\mathbf{V}(k)\right) = enc\left(\mathbf{V}(k)^*\right) \oplus enc\left(-f_{\boldsymbol{\sigma}_\mathbf{V}}(k)\right), \tag{4.72}$$
$$enc\left(\hat{\mathbf{U}}(k)\right) = enc\left(\hat{\mathbf{U}}(k)^*\right) \oplus enc\left(-f_{\boldsymbol{\sigma}_{\hat{\mathbf{U}}}}(k)\right), \tag{4.73}$$
$$enc\left(\hat{\mathbf{V}}(k)\right) = enc\left(\hat{\mathbf{V}}(k)^*\right) \oplus enc\left(-f_{\boldsymbol{\sigma}_{\hat{\mathbf{V}}}}(k)\right), \tag{4.74}$$
$$enc\left(\nabla\mathbf{U}_{\text{agg}}(k)\right) = enc\left(\nabla\mathbf{U}_{\text{agg}}(k)^*\right) \oplus enc\left(-f_{\boldsymbol{\sigma}_{\nabla\mathbf{U}}}(k)\right), \tag{4.75}$$
$$enc\left(\nabla\mathbf{V}_{\text{agg}}(k)\right) = enc\left(\nabla\mathbf{V}_{\text{agg}}(k)^*\right) \oplus enc\left(-f_{\boldsymbol{\sigma}_{\nabla\mathbf{V}}}(k)\right). \tag{4.76}$$

These resulting unmasked encrypted vectors $enc\left(\mathbf{U}(k)\right)$, $enc\left(\mathbf{V}(k)\right)$, $enc\left(\hat{\mathbf{U}}(k)\right)$, and $enc\left(\hat{\mathbf{V}}(k)\right)$ represent the updated profiles after epoch k. The unmasked encrypted aggregated gradients $enc\left(\nabla\mathbf{U}_{\text{agg}}(k)\right)$ and $enc\left(\nabla\mathbf{V}_{\text{agg}}(k)\right)$ are used by RecSys to check for convergence in Steps (19) and (20).

(19) Using the encrypted aggregated gradients from Step (18), RecSys determines convergence by ensuring the squared Euclidean norm of $\nabla\mathbf{U}_{\text{agg}}(k)$ is below its threshold $\omega_\mathbf{U}$, and the squared Euclidean norm of $enc\left(\nabla\mathbf{V}_{\text{agg}}(k)\right)$ is below its threshold $\omega_\mathbf{V}$.

RecSys obtains the element-wise squares of the aggregated gradient vectors. Let $\boldsymbol{\nabla}^2_\mathbf{U}$ and $\boldsymbol{\nabla}^2_\mathbf{V}$ represent these vectors of squared components:

$$\boldsymbol{\nabla}^2_\mathbf{U} := \nabla\mathbf{U}_{\text{agg}}(k) \odot \nabla\mathbf{U}_{\text{agg}}(k), \tag{4.77}$$
$$\boldsymbol{\nabla}^2_\mathbf{V} := \nabla\mathbf{V}_{\text{agg}}(k) \odot \nabla\mathbf{V}_{\text{agg}}(k). \tag{4.78}$$

Next, RecSys generates two random mask vectors, $\boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{U}} \in \mathbb{Z}^{dn}$ and $\boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{V}} \in \mathbb{Z}^{dm}$, initiating the masking procedure. RecSys then computes the masked versions of the squared gradient component vectors:

$$\boldsymbol{\nabla}^{2\,*}_\mathbf{U} := \boldsymbol{\nabla}^2_\mathbf{U} + \boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{U}}, \tag{4.79}$$
$$\boldsymbol{\nabla}^{2\,*}_\mathbf{V} := \boldsymbol{\nabla}^2_\mathbf{V} + \boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{V}}. \tag{4.80}$$

This is achieved homomorphically:

$$enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{U}\right) = \left(enc\left(\nabla\mathbf{U}_{\text{agg}}(k)\right) \otimes enc\left(\nabla\mathbf{U}_{\text{agg}}(k)\right)\right) \oplus enc\left(\boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{U}}\right), \tag{4.81}$$
$$enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{V}\right) = \left(enc\left(\nabla\mathbf{V}_{\text{agg}}(k)\right) \otimes enc\left(\nabla\mathbf{V}_{\text{agg}}(k)\right)\right) \oplus enc\left(\boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{V}}\right). \tag{4.82}$$

RecSys also computes the sum of the elements for each mask vector:

$$\Sigma^\sigma_\mathbf{U} = \sum_{p=1}^{dn} (\boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{U}})_p, \quad \text{and} \quad \Sigma^\sigma_\mathbf{V} = \sum_{p=1}^{dm} (\boldsymbol{\sigma}_{\boldsymbol{\nabla}^2_\mathbf{V}})_p. \tag{4.83}$$

Then, it prepares the comparison values for the CSP:

$$s_U = \omega_\mathbf{U} + \Sigma^\sigma_\mathbf{U}, \quad \text{and} \quad s_V = \omega_\mathbf{V} + \Sigma^\sigma_\mathbf{V}. \tag{4.84}$$

RecSys sends the ciphertexts $enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{U}\right)$, $enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{V}\right)$ and the plaintext scalars s_U and s_V to the CSP.

(20) The CSP receives $enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{U}\right)$, $enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{V}\right)$, s_U and s_V from RecSys. First, the CSP decrypts the received ciphertexts to obtain the plaintext masked vectors of squared gradient components:

$$\boldsymbol{\nabla}^{2\,*}_\mathbf{U} = dec\left(enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{U}\right)\right), \tag{4.85}$$
$$\boldsymbol{\nabla}^{2\,*}_\mathbf{V} = dec\left(enc\left(\boldsymbol{\nabla}^{2\,*}_\mathbf{V}\right)\right). \tag{4.86}$$

Next, CSP computes the sum of all elements for each of these plaintext masked vectors:

$$s'_U = \sum_{p=1}^{dn} (\boldsymbol{\nabla}^{2\,*}_{\mathbf{U}})_p, \quad \text{and} \quad s'_V = \sum_{p=1}^{dm} (\boldsymbol{\nabla}^{2\,*}_{\mathbf{V}})_p. \tag{4.87}$$

Note that $s'_U = (\sum_{p=1}^{dn} (\nabla \mathbf{U}_{\text{agg}}(k) \odot \nabla \mathbf{U}_{\text{agg}}(k))_p) + \Sigma^{\sigma}_{\mathbf{U}}$, which is effectively $\|\nabla \mathbf{U}_{\text{agg}}(k)\|^2 + \Sigma^{\sigma}_{\mathbf{U}}$. The CSP then performs the comparisons $s_U \geq s'_U$ and $s_V \geq s'_V$. These checks determine if $\|\nabla \mathbf{U}_{\text{agg}}(k)\|_2^2 \leq \omega_{\mathbf{U}}$ and $\|\nabla \mathbf{V}_{\text{agg}}(k)\|^2 \leq \omega_{\mathbf{V}}$, respectively. Let $b_U = (s_U \geq s'_U)$ and $b_V = (s_V \geq s'_V)$ be the boolean outcomes. The CSP sends the boolean pair (b_U, b_V) back to RecSys. If both b_U and b_V are true, the algorithm is considered to have converged, and RecSys proceeds to the Recommendation Phase. Otherwise, RecSys increments the epoch counter $k \leftarrow k + 1$ and returns to Step (9) to perform another learning iteration.

5

Design of a Privacy-Preserving Course Recommendation System

The previous chapter described a privacy-preserving matrix factorization protocol based on the work of Kim et al. We now pivot from the abstract cryptographic mechanism to a concrete application: a privacy-preserving course recommendation system for higher education. This adaptation presents us with two main challenges. The first challenge involves the contextual adaptation of this algorithm: we define the motivations of the involved parties and design the implementation of the protocol within the university setting.

The second challenge is to address the gap in the research: existing encryption-based private matrix factorization algorithms, including the protocol by Kim et al., do not incorporate baseline predictors (biases) into their models. While this omission simplifies the explanation of an already complex cryptographic mechanism, it significantly reduces the system's practical use, leaving private systems less accurate than their non-private matrix factorization counterparts. Integrating these biases, though trivial in plaintext, presents a substantial challenge in the cryptographic domain, as it requires careful protocol adjustments to preserve formal privacy guarantees.

We introduce a technique of data centering and vector augmentation that allows for the full integration of the global mean, student-specific, and course-specific biases into the cryptographic protocol. Our method achieves this without requiring large structural changes to the underlying algorithm, thereby preserving the protocol's formal security guarantees.

This chapter directly addresses the research question by detailing the design of our system and formalizing the system model for course recommendations. We then present our novel bias integration method, and detail its implementation within the protocol. The result is a blueprint for a system that is not only private by design, but also contextually relevant and able to achieve the same accuracy as its non-private counterpart. Furthermore, by cleanly separating systemic biases from the core interaction model, our design allows us to use these interactions to provide more nuanced recommendations moving beyond mere grade predictions and encouraging academic discovery.

5.1. System Model and Problem Formulation

Our course recommendation system is based on the collaborative filtering approach. Recall the setup of typical collaborative filtering recommender systems: Users provide explicit ratings for items. Then, based on these ratings, the system estimates the ratings that the user would give to the items they haven't yet rated. A few items with the best estimated scores are then recommended to each user.

The course recommendation problem is solved by modeling students as 'users' and courses as 'items', with the goal of finding a set of courses that can be recommended to each student. Rather than relying on explicit student ratings of courses, which are often sparse and require active solicitation, our model leverages a more objective data source already available within educational institutions: final

course grades. This method assumes that these grades are an indication of students' success in each particular course. The same approach has been used by Cakmak [77] and Thanh-Nhan, Nguyen, and Thai-Nghe [16] in their course recommendation systems. This design choice ensures that recommendations are based on demonstrated academic performance and merit, aligning the system's output with key educational objectives, which include not only predicting academic performance but also helping students discover areas of personal aptitude.

A central challenge in this domain is the 'cold start' problem, where the system cannot generate personalized recommendations for new students who lack a history of course grades. To address this, the model incorporates students' pre-university academic records. Each course that the student took in high school can be included as an item in the recommender system, with its obtained grade used as the rating. This ensures that all students have some kind of initial information for the first course recommendations. An example of how the input of such a system would look is shown in Figure 5.1.

Grading matrix		Pre-university courses		University courses		
		Course 1	**Course 2**	**Course 3**	**Course 4**	**Course 5**
Student	**Student 1**	9	6	8	7	**?**
	Student 2	5	7	**?**	3	6
	Student 3	6	5	5	**?**	9
	Student 4	10	7	10	7	6
	Student 5	10	6	8	**?**	**?**
	Student 6	6	8	**?**	**?**	**?**

Figure 5.1: An illustrative input grading matrix for the matrix factorization-based course recommendation system. Each row corresponds to a specific student, and each column represents a course. The matrix is populated with students' final grades (ranging from 0 to 10). Both pre-university (high school) and university records are included. The entries marked with a question mark ('?') represent the unknown grades for courses a student has not yet taken; these are the values the recommendation system is designed to predict.

5.1.1. Parties and Architecture

The system architecture adopts the two-server model from Kim, Kim, Koo, *et al.* [24], which operates under two key security assumptions:

- **Honest-but-curious threat model**, where each party correctly follows the protocol but may attempt to infer sensitive information from the data it processes,
- **Non-collusion assumption**, meaning the two core servers are trusted not to combine their information to compromise privacy.

The system involves the following parties:

- **The Student (User):** The data subject, who chooses to provide their academic history (grades) to receive personalized course recommendations. The student's primary goal is to obtain useful guidance while ensuring their sensitive grade data remains protected.
- **The University (Data Controller):** As the service coordinator, the university aims to improve student outcomes, reduce dropout rates, and optimize resource allocation. It is ethically and legally bound to protect student privacy, recognizing that doing so builds the trust necessary for wider adoption and more effective outcomes. With student consent, the university utilizes its securely stored student records to provide the recommendations.
- **Educational Technology Provider (Recommender System):** The entity that executes the recommendation algorithm. This service manages the encrypted model parameters (the profiles and biases) and orchestrates the overall protocol flow. It operates independently of the Crypto Service Provider and never has access to unencrypted student grades or the profiles used for recommendation.
- **The Crypto Service Provider (CSP):** A trusted server, operated by a neutral third party (e.g., a governmental institute), ensuring its independence. This server generates the cryptographic keys, distributes the public keys and locally stores the secret keys without revealing them to anyone. It

participates in the algorithm during data masking operations and never has access to any private information.

The interaction between these parties follows the privacy-preserving matrix factorization protocol explained in Chapter 4. The main difference in this educational context is that the grades (ratings) are submitted by the university instead of involving each student separately.

5.2. The Imperative of Bias Terms in Matrix Factorization

A pure matrix factorization model, where $\hat{r}_{ij} = \mathbf{u}_i^T \mathbf{v}_j$, while mathematically elegant, is suboptimal for real-world recommendation tasks. Seminal work from the Netflix Prize competition showed that simple, systemic tendencies explain a significant portion of the variance in ratings data. These tendencies, or biases, include users who consistently give high ratings and items that are generally rated poorly. While a pure matrix factorization model could implicitly learn these effects, it is more effective to model them explicitly. This allows the latent factors to focus on the more nuanced user-item interactions [25]. This leads to the widely-used biased Matrix Factorization model, formulated as:

$$\hat{r}_{ij} = \mu + b_i + b_j + \mathbf{u}_i^T \mathbf{v}_j. \tag{5.1}$$

In the context of course recommendations, these terms have clear and powerful interpretations that capture fundamental academic dynamics:

- μ (Global Mean): The average grade across all students in all courses at the university. This serves as the baseline academic performance.
- b_j (Item Bias / Course Bias): The intrinsic difficulty or grading tendency of a specific course. A notoriously challenging course would naturally have a strong negative bias ($b_j < 0$), while a course known for lenient grading would have a positive bias ($b_j > 0$).
- b_i (User Bias / Student Bias): The general academic standing or aptitude of a student, independent of specific subjects. A student who consistently achieves high marks across their curriculum will have a positive bias ($b_i > 0$), whereas one who generally receives lower grades will have a negative bias ($b_i < 0$).

During the learning phase, the global mean μ is treated as a fixed, pre-computed value. In contrast, the student and course biases, b_i and b_j, are treated as learnable parameters, just like the latent factors $\mathbf{u}_i$ and $\mathbf{v}_j$. This approach provides greater model flexibility, allowing the algorithm to iteratively adjust the biases to best capture the systemic tendencies specific to each student and course, in contrast to using pre-determined values [25].

By explicitly modeling these baseline effects, the latent factor interaction term, $\mathbf{u}_i^T \mathbf{v}_j$, is freed to capture the more subtle dynamics between the academic strengths of the student and the unique characteristics of the course. Omitting these bias terms is therefore not a minor simplification, but a major limitation of the model's performance. Their inclusion is imperative for building a genuinely useful course recommendation system.

5.3. A Novel Method for Bias Integration

Given the matrix factorization protocol that models the ratings as a simple dot product between the users and items (i.e., $r_{ij} = \mathbf{u}_i^T \mathbf{v}_j$), the challenge is to adjust this algorithm so that it can compute the full biased model (Equation 5.1). Directly adding new parameters and update rules would require a redesign of the complex HE protocol.

We propose an elegant strategy that resolves this by reframing the input data and model vectors, thereby requiring no modifications to the protocol's core cryptographic operations. Our method consists of two parts:

1. **Data Centering:** We introduce a cryptographic technique to center the rating data, effectively accounting for the global mean bias μ.
2. **Vector Augmentation:** We incorporate the student and course biases, b_i and b_j, by augmenting the student and course profile vectors, $\mathbf{u}_i$ and $\mathbf{v}_j$.

Crucially, these adjustments are performed entirely within the encrypted domain. Our solution is an enhancement of the protocol from Chapter 4 and does not assume that any party, such as the university, has plaintext access to the full rating set. This approach ensures that the end-to-end privacy guarantees of the system are maintained.

5.3.1. Handling the Global Mean (μ) via Data Centering

The first bias term, the global mean μ, represents the average of all known ratings. Note that it is possible to eliminate the need for this term by pre-computing the mean value and subtracting it from every rating, a process known as data centering. The model would then be trained on these centered ratings, effectively reducing the prediction formula to:

$$\hat{r}_{ij} = b_i + b_j + \mathbf{u}_i^T \mathbf{v}_j. \tag{5.2}$$

The challenge in our encrypted protocol is that no single party has access to the plaintext ratings to compute μ.

To overcome this challenge, we introduce a method that performs data centering on the masked data. This is achieved by adjusting the rating upload phase of the protocol. The core idea is for the CSP to compute the mean of the masked ratings and center the data using this value. Subsequently, RecSys corrects for the influence of the masking noise on this computed mean. The two adjusted steps of the original protocol are detailed below. Note that changes made to the original equations are denoted in blue.

Learning Phase Modifications

(6) CSP receives $r^{(\text{all rating set})*}$. They decrypt all masked ratings using the secret key sk_{AHE}, obtaining a set of triplets $\{(i, j, \overline{r_{ij} + \sigma_{ij}}) : (i, j) \in \mathcal{M}\}$.

First, CSP calculates the average value of all these masked ratings, which we denote as μ^*:

$$\mu^* := \frac{\sum_{(i,j)\in\mathcal{M}} \overline{r_{ij} + \sigma_{ij}}}{|\mathcal{M}|}. \tag{5.3}$$

This μ^* represents the mean of the ratings combined with the mean of the random masks:

$$\mu^* = \frac{\sum(r_{ij} + \sigma_{ij})}{|\mathcal{M}|} = \frac{\sum r_{ij}}{|\mathcal{M}|} + \frac{\sum \sigma_{ij}}{|\mathcal{M}|} = \mu + f_\mu, \tag{5.4}$$

where f_μ can be seen as the unmasking term.

The masked ratings are sorted, centered, negated, and combined to obtain a packed dM-dimensional vector:

$$\mathbf{r}^* = (\|_{(i,j\in\mathcal{M})} \overbrace{(\overline{-(r_{ij} + \sigma_{ij} - \mu^*)}, 0, \ldots, 0)}^{d}). \tag{5.5}$$

This vector, containing all centered ratings, is in the format that is required for the future protocol. However, these ratings are still masked. The array is encrypted using the public key pk_{HE}, and the encryption $enc_{\text{pk}_{\text{HE}}}(\mathbf{r}^*)$ is sent to RecSys, who will unmask the ratings, obtaining the encryption of $\mathbf{r}$ in the next step.

(7) RecSys must now remove the effect of its random masks σ_{ij} from the encrypted vector. RecSys computes the average of its own noise values, f_μ:

$$f_\mu := \frac{\sum_{(i,j)\in\mathcal{M}} \overline{\sigma_{ij}}}{|\mathcal{M}|}. \tag{5.6}$$

The influence can then be computed as:

$$f_\sigma := (\|_{(i,j\in\mathcal{M})} \overbrace{(\overline{-(\sigma_{ij} - f_\mu)}, 0, \ldots, 0)}^{d}). \tag{5.7}$$

To obtain the final packed and centered rating vector $\mathbf{r}$, f_σ is negated, encrypted, and added to the received array using the additive property of HE.

$$enc_{\mathsf{pk_{HE}}}(\mathbf{r}^*) \oplus enc_{\mathsf{pk_{HE}}}(-f_\sigma) \tag{5.8}$$

$$= enc_{\mathsf{pk_{HE}}}\left((\|_{(i,j)\in\mathcal{M}} \overbrace{(-(r_{ij}+\sigma_{ij}-\mu^*)}^{d}, 0, \ldots, 0)) - (\|_{(i,j)\in\mathcal{M}} \overbrace{(-(\sigma_{ij}-f_\mu)}^{d}, 0, \ldots, 0)) \right) \tag{5.9}$$

$$= enc_{\mathsf{pk_{HE}}}\left((\|_{(i,j)\in\mathcal{M}} \overbrace{(-(r_{ij}-\mu)}^{d}, 0, \ldots, 0)) \right) \tag{5.10}$$

$$= enc_{\mathsf{pk_{HE}}}(\mathbf{r}) . \tag{5.11}$$

The homomorphic addition precisely removes the mask and its influence on the mean. This leaves the correctly centered ratings encrypted under $\mathsf{pk_{HE}}$, yielding the target vector $enc_{\mathsf{pk_{HE}}}(\mathbf{r})$.

The remainder of the algorithm proceeds as originally designed, but now operates on the centered rating data. During the final recommendation phase, it is not necessary to add the global mean μ back to the predicted scores. The purpose of the system is to generate a top-N list of recommended courses, which depends only on the relative ordering of the predicted ratings. Since subtracting a constant (μ) from all ratings does not change this ordering, the centered predictions are sufficient.

5.3.2. Handling Biases (b_i, b_j) via Vector Augmentation

To integrate the student and course biases, we redefine the existing profile vectors:

1. The latent vectors are augmented from dimension d to $d+2$.
2. These augmented vectors, $\tilde{\mathbf{u}}_i \in \mathbb{R}^{d+2}$ (for student i) and $\tilde{\mathbf{v}}_j \in \mathbb{R}^{d+2}$ (for course j), are structured as follows:

$$\tilde{\mathbf{u}}_i := \begin{bmatrix} \mathbf{u}_i \\ 1 \\ b_i \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{v}}_j := \begin{bmatrix} \mathbf{v}_j \\ b_j \\ 1 \end{bmatrix}, \tag{5.12}$$

   where $\mathbf{u}_i$ and $\mathbf{v}_j$ are the original d-dimensional latent vectors, b_i and b_j are the learnable scalar bias parameters, and the ‘1‘s are fixed constants.
3. The optimization algorithm learns all components of the augmented vectors, but the constant 1s are enforced on each iteration by having the CSP reset their values, which nullifies any gradient-based updates to those positions.

This reframing allows the bias terms to be implicitly computed by taking the dot product of these two augmented vectors.

Verification. The dot product of these augmented vectors, $\hat{r}_{ij} = \tilde{\mathbf{u}}_i^T \tilde{\mathbf{v}}_j$, now reconstructs the desired model for the centered data:

$$\hat{r}_{ij} = \begin{bmatrix} \mathbf{u}_i^T & 1 & b_i \end{bmatrix} \begin{bmatrix} \mathbf{v}_j \\ b_j \\ 1 \end{bmatrix} = (\mathbf{u}_i^T \mathbf{v}_j) + (1 \cdot b_j) + (b_i \cdot 1) = b_i + b_j + \mathbf{u}_i^T \mathbf{v}_j. \tag{5.13}$$

This demonstrates that our method of centering and augmentation allows the full, accurate Biased MF model to be computed using only the dot product operation for which the cryptographic protocol was originally designed.

This approach can be implemented by making the following adjustments to the original algorithm from Chapter 4. The changes made to the original equations are denoted in blue.

Setup Phase Modifications

(2) RecSys determines and publishes public parameters. The profile vector dimension is now set to $d' = d + 2$. RecSys publishes public parameters $(d', n, m, \alpha, \beta)$.

Next, RecSys must initialize the augmented vectors correctly, ensuring the constant '1's are in their proper positions.

Learning Phase Modifications

(8) RecSys generates the initial profile vectors. These are now the augmented $(d + 2)$-dimensional vectors $\tilde{\mathbf{u}}_i(0)$ and $\tilde{\mathbf{v}}_j(0)$.

For each user $i \in [n]$ and item $j \in [m]$, RecSys initializes:

$$\tilde{\mathbf{u}}_i(0) := \begin{bmatrix} \mathbf{u}_i(0) \\ \overline{1} \\ b_i(0) \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{v}}_j(0) := \begin{bmatrix} \mathbf{v}_j(0) \\ b_j(0) \\ \overline{1} \end{bmatrix}, \tag{5.14}$$

where $\mathbf{u}_i(0) \in \mathbb{R}^d$ are initialized as random ℓ_2-norm vectors. The scalars $b_i(0), b_j(0)$ are set to 0. The terms $\overline{1}$ are constants representing the value '1' in fixed-point arithmetic. These augmented vectors are converted to fixed-point representation and then used to construct the four initial packed vectors, which are now of dimension $(d+2)M$. These are then encrypted.

The most important changes occur during the iterative update steps. The CSP must enforce the constant '1's, and RecSys must account for this action during unmasking.

(16) The CSP decrypts the masked update vectors, aggregates them, and applies scaling. After computing the updated (but still masked) profile vectors for each user and item, the CSP must intervene to enforce the constant '1's before re-packing and re-encrypting.

After computing $\mathbf{u}_i(k)^*$ and $\mathbf{v}_j(k)^*$, but before constructing the packed vectors, the CSP adjusts specific elements in masked profile vectors for all $i \in [n]$ and $j \in [m]$:

$$(\mathbf{u}_i(k)^*)_{d+1} = 2^\alpha \quad \text{and} \quad (\mathbf{v}_j(k)^*)_{d+2} = 2^\alpha. \tag{5.15}$$

The value is set to 2^α, which is the fixed-point representation of '1' with precision α. This operation overwrites whatever value was computed for that component, effectively zeroing out its gradient and resetting it to '1'. The CSP then proceeds to pack these modified vectors and encrypt them as described in the original protocol.

(18) RecSys receives the encrypted updated profiles from the CSP and must unmask them. Since the CSP overwrote the values at the constant positions, these components are no longer masked. Therefore, RecSys must adjust its unmasking terms to avoid removing noise that is not present in these fixed components.

When constructing the final unmasking vectors $f_{\boldsymbol{\sigma}_\mathbf{U}}(k)$ and $f_{\boldsymbol{\sigma}_\mathbf{V}}(k)$, RecSys must zero out the components corresponding to the fixed '1's. Given $(\Delta\boldsymbol{\sigma}_\mathbf{U}(k))_i$ and $(\Delta\boldsymbol{\sigma}_\mathbf{V}(k))_j$, the masking vectors generated in Step (15), RecSys modifies specific elements in these vectors for all $(i, j) \in \mathcal{M}$:

$$(\boldsymbol{\sigma}_\mathbf{U}(k)_{ij})_{d+1} = 0 \quad \text{and} \quad (\boldsymbol{\sigma}_\mathbf{V}(k)_{ij})_{d+2} = 0. \tag{5.16}$$

RecSys then uses these modified aggregated masks to construct the final unmasking terms $f_{\boldsymbol{\sigma}_\mathbf{U}}(k)$, $f_{\boldsymbol{\sigma}_\mathbf{V}}(k)$, $f_{\boldsymbol{\sigma}_{\tilde{\mathbf{U}}}}(k)$, and $f_{\boldsymbol{\sigma}_{\tilde{\mathbf{V}}}}(k)$. By zeroing out these specific entries, RecSys ensures that no mask is subtracted from the fixed '1' components during the homomorphic unmasking operation, preserving their integrity.

The rest of the protocol, including the convergence check, proceeds with the adjusted $(d+2)$-dimensional vectors without further changes.

The proposed methods of data centering and vector augmentation allow the complete biased Matrix Factorization model to be incorporated into the privacy-preserving protocol, thereby overcoming its architectural limitation to simple dot-product computations. The following chapter will provide theoretical and empirical validation of its security, accuracy, and performance.

5.3.3. Beyond Grade Prediction: Discovering Individual Talent

The successful separation of bias terms from the core latent factor model does more than just improve predictive accuracy; it expands the system's purpose. Following the work of Bell and Koren [17], our approach of incorporating bias terms during training allows the model to absorb systematic tendencies. This ensures that the latent factors are free to model the pure, underlying user-item compatibility. Our architecture can therefore answer two distinct questions for a student:

1. *"In which course am I most likely to achieve a high grade?"*
2. *"For which course do I possess a unique, perhaps undiscovered, aptitude?"*

The first question is answered by the full prediction model from Equation 5.2: $\hat{r}_{ij} = b_i + b_j + \mathbf{u}_i^T \mathbf{v}_j$. This score represents the predicted outcome, accounting for the student's general academic standing (b_i) and the course's baseline difficulty (b_j). The second, more nuanced question is addressed by isolating the interaction term, $\mathbf{u}_i^T \mathbf{v}_j$. This term is a proxy for the synergy between a student's unique profile and a course's specific characteristics.

This functional separation is a direct consequence of our bias integration method. By explicitly modeling and learning the biases, we enable the system to generate two distinct top-N lists for each student: one optimized for predicted performance and another for discovering personal talent. For example, a student might be discouraged from a difficult course (with a negative b_j), but a high interaction term could reveal that they are exceptionally well-suited for it, suggesting a potential passion that would otherwise be overlooked. This transforms the system from a simple grade predictor into a more nuanced tool for personalized academic guidance.

However, while our approach successfully separates a student's specific aptitude from general course difficulty, it operates on data that is already influenced by student choices. Our approach cannot, by itself, correct for the selection bias where popular courses are overrepresented in the data, a feedback loop effect demonstrated by Mansoury, Abdollahpouri, Pechenizkiy, *et al.* [78]. A truly debiased recommendation for 'hidden talent' would require causal inference methods to model the probability that a student would select a course in the first place.

6

Analysis

Having established the design of our privacy-preserving course recommendation system in the previous chapter, this chapter turns to its evaluation. We assess the proposed system from three critical perspectives: security, efficiency, and empirical performance.

First, in section 6.1, we provide a thorough security analysis to demonstrate that our novel modifications do not compromise the protocol's foundational privacy guarantees. Second, in section 6.2, we conduct a theoretical efficiency analysis to quantify the computational and communication overhead introduced by our enhancements. Third, in section 6.3, we present an empirical evaluation using a real-world dataset, demonstrating the significant gains in recommendation accuracy and model convergence that our system achieves.

Building on these results, we conclude in section 6.4 with a cost-benefit analysis, weighing the true computational cost against the resulting accuracy. This will demonstrate that our enhancements lead to a superior model, validating that our system is not only secure by design but also powerful and efficient enough for practical application.

6.1. Security Guarantees

The security of our enhanced protocol is based on the same assumptions as the original protocol by Kim et al. [24]: the honest-but-curious adversarial model and non-collusion between the Recommender System (RecSys) and the Crypto-Service-Provider (CSP). In this model, parties follow the protocol specification faithfully but may attempt to infer additional information from the data they observe. The non-collusion assumption guarantees that the CSP and RecSys do not share their secret information (cryptographic keys and masking values, respectively).

Our contributions, Data Centering and Vector Augmentation, are designed as enhancements that integrate into the existing cryptographic protocol. In the following subsections, we analyze each modification to prove that they introduce no new vulnerabilities and that the system's original security guarantees are preserved.

6.1.1. Security of Data Centering

Our data centering method modifies the rating upload phase to subtract the global mean μ from all ratings. This is achieved through a two-step interaction: first, the CSP computes a mean on the masked ratings and centers the data using this value. Second, RecSys homomorphically corrects for the influence of the masks on the centered data.

- **CSP's View:** The CSP receives the set of masked ratings $\{(i, j, r_{ij} + \sigma_{ij})\}$, which is identical to the data it would receive in the original protocol. The new operation it performs is computing the mean of these values, which we denote as μ^*. Since all individual masks are randomly generated and only known by RecSys, the computed value μ^* reveals no information about the true mean μ. Consequently, using μ^* to center the masked ratings also leaks no information about the true

centered ratings.

- **RecSys's View:** RecSys receives an encrypted vector of the masked and centered ratings from the CSP. It then computes a correction term, f_σ, derived from the mean of its own secret masks. By homomorphically subtracting this correction term, the final encrypted rating vector, enc(r') can be obtained. Throughout this process, RecSys only ever manipulates encrypted data. Without the CSP's secret key, it cannot decrypt the vector to learn the underlying grades. Therefore, no new information is made available to RecSys.

In summary, the data centering modification reveals no new plaintext information to either party. The security remains strictly dependent on the non-collusion assumption, which, if violated, would break the security of the original protocol as well.

6.1.2. Security of Vector Augmentation

The vector augmentation method integrates student and course biases (b_j, b_j) by expanding the profile vectors and enforcing constant values in specific positions.

- **CSP's View:** The CSP's primary new action is to overwrite specific components of the updated (but still masked) profile vectors during the learning phase. Specifically, it sets $(\mathbf{u}_i(k)^*)_{d+1} = 2^\alpha$ and $(\mathbf{v}_j(k)^*)_{d+2} = 2^\alpha$. This action does not introduce a security risk. The value 2^α is the public, fixed-point representation of the constant '1', a part of the public model structure. The CSP performs this overwrite on masked vectors, meaning it does not know the true values of the other components (the latent factors and biases). Enforcing a public constant on masked data reveals no new information.
- **RecSys's View:** RecSys is aware that the protocol requires the CSP to reset certain vector components. However, because the system uses a probabilistic encryption scheme, the resulting ciphertexts for the constant '1' are computationally indistinguishable from the ciphertexts of any other value. RecSys's only action is to adjust its own secret unmasking terms to ensure that no random mask is subtracted from these constant components—a procedural adjustment based on public knowledge of the protocol, not a leakage of secret data. RecSys still only receives encrypted profile vectors from the CSP and cannot decrypt them to obtain any sensitive information.

In conclusion, the vector augmentation method is secure under the same assumptions as the original protocol. The modifications do not introduce new information leakage, as they consist of procedural adjustments performed on masked or encrypted data. This technique allows for the full biased Matrix Factorization model to be computed within the encrypted domain without compromising the system's privacy guarantees.

6.2. Theoretical Efficiency Analysis

In this section, we analyze the computational and communication complexity of the original protocol (introduced in Chapter 4) and compare it to our adapted version presented in Chapter 5. Since both algorithms share the same asymptotic complexity in Big-O notation, such a high-level analysis is insufficient for a practical comparison. The protocol's overall runtime is overwhelmingly dominated by its cryptographic components; operations like homomorphic encryption and ciphertext multiplication are several orders of magnitude more computationally expensive than simple plaintext arithmetic, making the cost of all other steps negligible. Therefore, to provide a more precise and meaningful evaluation of their relative efficiency, our analysis considers a detailed count of these dominant cryptographic operations. Note that our analysis doesn't include the initial setup phase which consists of generating and distributing the variables, as its complexity will depend on the specific implementation and is a one-time cost that is amortized over the entire learning process, making it non-dominant in practice. The efficiency analysis is based on the following variables:

- d: the dimension of the latent profile vectors.
- M: the total number of known ratings in the dataset $\mathcal{M}$.
- L: the number of available slots in a single HE ciphertext for data packing.
- n: the number of users.

- m: the number of items.
- K_{iter}: the total number of iterations in the Learning Phase.
- $|HE|$ and $|AHE|$: the bit-size of a single ciphertext under the FHE and AHE schemes, respectively.

To simplify the notation in the complexity formulas, we define the number of ciphertexts required for the main packed vectors as $\alpha := \lceil dM/L \rceil$ and for both of the aggregated gradient vectors as $\beta := \lceil dn/L \rceil + \lceil dm/L \rceil$.

Computational Complexity

The following tables present the total count of cryptographic operations required by each party. We differentiate between Additive Homomorphic Encryption (AHE), used in the upload phase, and Fully Homomorphic Encryption (FHE), used in the learning phase, as their performance can differ significantly. This detailed breakdown allows for a nuanced comparison and can inform the selection of optimal cryptographic schemes for a given implementation. Tables 6.1 and 6.2 summarize these counts. A more detailed step-by-step derivation can be found in Table 6.4.

Table 6.1: Count of Additive Homomorphic Encryption Operations per Party.

Operation	**Number of AHE Operations per Party**		
	Users (Total)	RecSys	CSP
Encryption	M	M	—
Addition	—	M	—
Decryption	—	—	M

Table 6.2: Count of Fully Homomorphic Encryption Operations per Party.

Operation	**Number of FHE Operations per Party**	
	RecSys	CSP
Encryption	$K_{\text{iter}} \cdot (10\alpha + 2\beta) + 5\alpha$	$K_{\text{iter}} \cdot (5\alpha + \beta) + \alpha$
Decryption	—	$K_{\text{iter}} \cdot (5\alpha + \beta)$
Addition	$K_{\text{iter}} \cdot (10\alpha + \beta) + \alpha$	—
Scalar Mult.	$K_{\text{iter}} \cdot (7\alpha)$	—
Multiplication	$K_{\text{iter}} \cdot (3\alpha + \beta)$	—

Communication Complexity

The communication overhead is also critical for evaluating the protocol's practicality, as it is primarily driven by the exchange of large ciphertexts. Table 6.3 summarizes the total data transferred over the entire execution of the protocol, assuming the learning phase runs for K_{iter} iterations. The costs are broken down by communication channel.

Table 6.3: Total Communication Complexity of the Protocol.

Channel	**Total Data Transferred**
Users $\rightarrow$ RecSys	$M \cdot \lvert AHE \rvert$
RecSys $\rightarrow$ CSP	$M \cdot \lvert AHE \rvert + K_{\text{iter}} \cdot (5\alpha + \beta) \cdot \lvert HE \rvert + K_{\text{iter}} \cdot$ 2 scalars
CSP $\rightarrow$ RecSys	$(\alpha + K_{\text{iter}} \cdot (5\alpha + \beta)) \cdot \lvert HE \rvert + K_{\text{iter}} \cdot$ 2 booleans

Comparison with Our Adapted Algorithm

The adaptation of the protocol to include mean and individual biases, as presented in Chapter 5, preserves the overall protocol structure. All cryptographic steps of the protocol remain the same; the only modification is an increase in the dimensionality of the profile vectors from d to $d + 2$ to accommodate the bias terms.

This change has a direct and predictable impact on the efficiency. As shown in the complexity formulas in Tables 6.2 and 6.3, the dominant costs are driven by the number of ciphertexts, which grows

Table 6.4: Detailed Theoretical Efficiency Analysis of the Protocol.

Phase	Step	Party	Computational Complexity	Communication Complexity
Rating Upload	4	Users	$M \times$ AHE Enc	$M \times \|AHE\|$ to RecSys
	5	RecSys	$M \times$ AHE Enc, $M \times$ AHE Add	$M \times \|AHE\|$ to CSP
	6	CSP	$M \times$ AHE Dec, $\alpha \times$ HE Enc	$\alpha \times \|HE\|$ to RecSys
	7	RecSys	$\alpha \times$ HE Enc, $\alpha \times$ HE Add	—
Learning (1 Iteration)	8 (Init)	RecSys	$4\alpha \times$ HE Enc	—
	9	RecSys	$\alpha \times$ (HE Mult, HE SM, HE Add)	—
	10	RecSys	$\alpha \times$ (HE Enc, HE Add)	$\alpha \times \|HE\|$ to CSP
	11	CSP	$\alpha \times$ HE Dec	—
	12	CSP	$\alpha \times$ HE Enc	$\alpha \times \|HE\|$ to RecSys
	13	RecSys	$\alpha \times$ (HE Enc, HE Add)	—
	14	RecSys	$2\alpha \times$ HE Mult, $6\alpha \times$ HE SM, $4\alpha \times$ HE Add	—
	15	RecSys	$4\alpha \times$ (HE Enc, HE Add)	$4\alpha \times \|HE\|$ to CSP
	16	CSP	$4\alpha \times$ HE Dec, $(4\alpha) \times$ HE Enc	$4\alpha \times \|HE\|$ to RecSys
	17	CSP	$\beta \times$ HE Enc	$\beta \times \|HE\|$ to RecSys
	18	RecSys	$(4\alpha + \beta) \times$ (HE Enc, HE Add)	—
	19	RecSys	$\beta \times$ (HE Mult, HE Enc, HE Add)	$\beta \times \|HE\|$ and 2 scalars to CSP
	20	CSP	$\beta \times$ HE Dec	2 booleans to RecSys (negligible)

proportionally with the dimension d. By replacing d with $d + 2$, the computation and communication complexities increase by at most $2/d$ relative to the original cost (i.e., multiplicative factor $(d + 2)/d$). Given that d is typically in the range of 10-100, increasing the complexities by $2/d$ represents a relatively small percentage increase.

While this adaptation introduces a slight computational burden, we will argue in the following section that the inclusion of bias terms significantly accelerates model convergence. This can lead to a substantial reduction in the required number of iterations, K_{iter}, potentially outweighing the modest increase in per-iteration cost and resulting in a more efficient protocol overall.

6.3. Experimental Evaluation

Our theoretical analysis established that incorporating bias terms into the cryptographic protocol introduces a modest and predictable computational overhead. This section now provides the empirical counterpart to that analysis, designed to validate a central claim of this thesis: that the inclusion of global mean and bias terms is a critical optimization that significantly enhances recommendation accuracy and accelerates model convergence.

While bias terms are a standard feature in many modern matrix factorization implementations, a rigorous, comparative analysis of their specific impact on convergence speed and final accuracy is surprisingly scarce in the literature, particularly for models optimized with batch gradient descent. This experiment is therefore designed to fill that gap by systematically quantifying the benefits of each component.

To isolate the impact of these architectural improvements from the complexities of the cryptographic protocol, we conduct our evaluation in the plaintext domain. This methodology is sound because our chosen cryptographic framework, Homomorphic Encryption (HE), performs exact computations on encrypted data. The HE protocol guarantees that model performance is identical in both plaintext and encrypted settings, unlike statistical privacy methods that operate by intentionally introducing noise. This allows us to measure the performance gains with high confidence that the results translate directly to the privacy-preserving implementation detailed in Chapter 5.

6.3.1. Methodology

This section details the empirical methodology used to quantify the performance gains from integrating bias terms into a matrix factorization model. We systematically compare four model variants to isolate the contribution of each architectural component.

Models for Comparison

To systematically isolate the effect of each component (the global mean and the individual user/item biases), we compare the performance of four distinct matrix factorization models. These models incrementally build from the standard algorithm used in the baseline protocol by Kim et al. to our fully enhanced model, allowing us to precisely measure the independent contribution of each improvement.

1. **MF-Base:** The standard matrix factorization model, learning only user and item latent factors (profiles). The predicted rating $\hat{r}_{ij}$ for user i and item j is modeled as:
$$\hat{r}_{ij} = u_i^T v_j. \tag{6.1}$$
2. **MF+Mean:** This model incorporates the global average rating (μ) as a baseline predictor, accounting for the overall rating tendency in the dataset. The prediction is:
$$\hat{r}_{ij} = \mu + u_i^T v_j. \tag{6.2}$$
3. **MF+Biases:** This model incorporates user-specific biases (b_i) and item-specific biases (b_j). The user bias captures a user's tendency to give higher or lower ratings than average, while the item bias captures an item's tendency to receive them. The prediction is:
$$\hat{r}_{ij} = b_i + b_j + u_i^T v_j. \tag{6.3}$$
4. **MF+Mean+Biases:** Our proposed full model, which integrates all components. This is the common and powerful variant of matrix factorization in non-private literature. The prediction formula is:
$$\hat{r}_{ij} = \mu + b_i + b_j + u_i^T v_j. \tag{6.4}$$

By comparing these four variants, we can directly measure the performance contribution of the global mean and the user/item biases, both individually and in combination.

Dataset

Given the lack of publicly available student grade datasets, we utilize the well-established `MovieLens-100k` dataset [79] as a benchmark. This dataset is a standard in the recommender systems literature and shares key characteristics with our target domain, such as a sparse user-item interaction matrix and explicit ratings (analogous to grades). The dataset comprises 100,000 ratings on a scale of 1 to 5, from 943 users on 1682 movies.

To ensure a robust and unbiased evaluation, the dataset was divided into training, validation, and test sets. We employed a user-stratified random split, allocating 80% of each user's ratings to the *training set*, 10% to the *validation set*, and the final 10% to the *test set*. This methodology prevents data leakage and provides a realistic assessment of the model's ability to generalize to unseen data for existing users. A fixed random seed was used for all splits to ensure reproducibility.

Evaluation Plan and Metrics

To evaluate model performance, we selected two complementary metrics, the formal definitions of which can be found in subsection 2.1.4. While both are important, they assess different aspects of the system's utility.

Our primary metric for measuring the practical success of the system is **Normalized Discounted Cumulative Gain (nDCG@10)**. We prioritize this metric because it directly reflects the real-world use case of our system. A student typically chooses from a known, limited set of available courses each semester. In this context, the quality of the ranked list presented to them is paramount. Unlike set-based metrics such as Precision or Recall, which are useful for open-ended content discovery, nDCG@k specifically evaluates the crucial ordering of recommendations. It rewards the model for placing the most promising courses at the top of the list, which is the most faithful measure of the system's utility to the student. We use a cutoff of $k = 10$ to simulate a student reviewing a "top-10" list of suggestions.

Our second metric is **Root Mean Squared Error (RMSE)**. While nDCG evaluates the final user-facing output, RMSE measures the underlying predictive accuracy of the model's grade predictions and serves a critical role during the training and optimization process.

The roles for each metric in our evaluation plan are therefore distinct:

- **RMSE** is used to guide model training and assess the final predictive accuracy. We minimize the RMSE on the *validation set* during hyperparameter optimization, and then use the RMSE on the *test set* to measure the final model's generalization performance.
- **nDCG@10** is used to assess the final ranking quality of the optimized models. It is computed exclusively on the held-out *test set*, providing a definitive measure of real-world performance.

Hyperparameter Optimization

The performance of a matrix factorization model is governed by its hyperparameters—settings that are not learned from the data but are configured before the training process begins. The choice of these hyperparameters critically influences both the final accuracy of the model and the efficiency of its training.

To conduct a fair comparison, one must find the optimal hyperparameters for each of the four architectures. A search focused solely on the best achievable accuracy, however, would be insufficient. It would fail to test our hypothesis: that the inclusion of bias terms not only improves accuracy but also significantly accelerates model convergence. We must therefore analyze the interaction between predictive accuracy and training efficiency.

This leads us to frame the analysis as a multi-objective optimization problem. We aim to find the set of hyperparameters that simultaneously minimize two competing objectives:

1. **Validation RMSE:** The lowest Root Mean Squared Error achieved on the validation set.
2. **Epochs to Converge:** The number of training epochs required to reach that best validation RMSE, representing the point of optimal performance before overfitting begins.

The solution to such a problem is not a single "best" set of hyperparameters but a set of optimal configurations known as the Pareto optimal front. Each point on this front represents a configuration that is non-dominated, meaning no other configuration is superior on both objectives simultaneously. Analyzing this front allows us to draw robust conclusions about each model's performance trade-offs.

To automate the discovery of this front, we employ the `Optuna` framework [80]. Unlike exhaustive grid search, Optuna performs a guided search where, based on the results of past experiments, it intelligently proposes new hyperparameter combinations to test. A single trial in this process is defined as one complete training-and-evaluation cycle:

1. Optuna selects a new set of hyperparameters from the defined search space.
2. A model is trained from scratch using these parameters, running for a maximum of 500 epochs with an early stopping patience of 25. This patience mechanism is effective because batch gradient descent produces a very smooth, U-shaped validation error curve, making the optimal point unambiguous.
3. The model's performance (its best Validation RMSE and the Epoch at which it occurred) is recorded and returned to Optuna.

This process was repeated for 700 trials for each of the four models. The search space for the hyperparameters was defined as follows:

- **Latent Factor Dimension (d):** Integer from `5` to `40`.
- **Learning Rate (η):** Log-uniform float from `1e-5` to `1e-2`.
- **Learning Rate for Biases (η_{bias}):** Log-uniform float from `1e-5` to `1e-2` (for relevant models).
- **Regularization Strength (λ):** Log-uniform float from `1e-4` to `50.0`.

6.3.2. Hyperparameter Optimization Results: Pareto Analysis

The results of the multi-objective hyperparameter optimization are visualized in Figure 6.1. This figure plots the Pareto optimal front for each of the four model architectures. The front itself—the boundary line on the graph—represents the set of non-dominated solutions discovered, illustrating the limit of performance for that architecture. This line shows the best achievable trade-off between validation

RMSE (x-axis) and the epochs required for convergence (y-axis). A front is considered superior if it is positioned lower and to the left, representing better accuracy achieved in fewer training iterations.

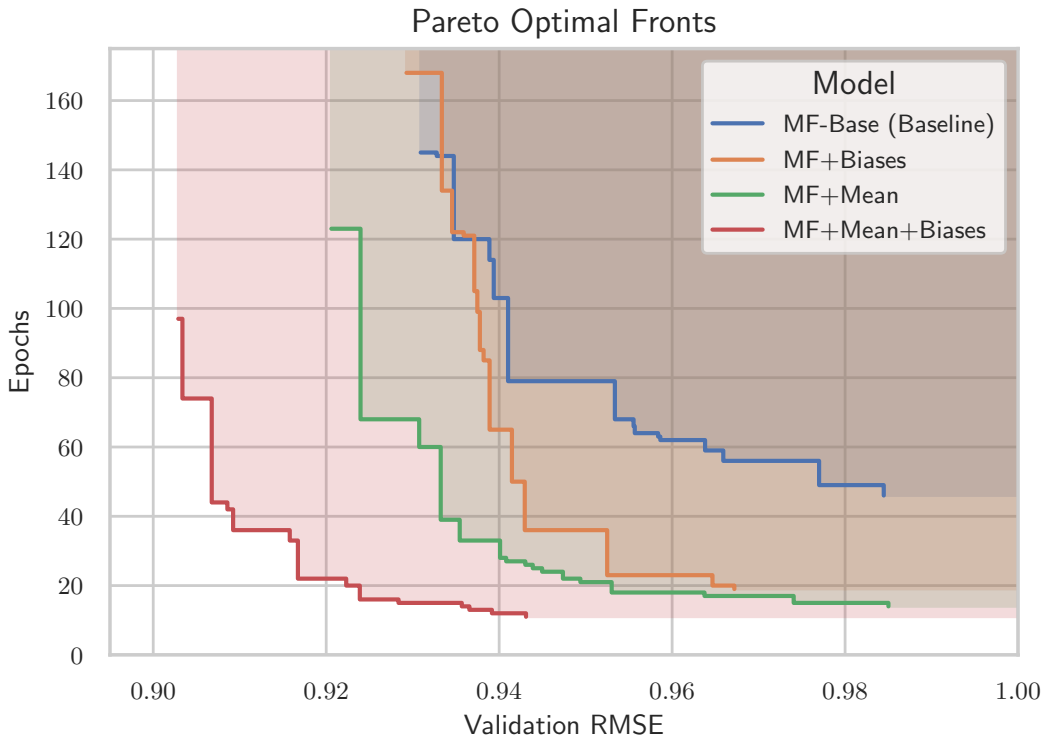

Figure 6.1: Pareto fronts for the four models, showing the trade-off between validation RMSE and epochs to converge. The shaded areas indicate the regions where each model is not dominated by any other.

The analysis reveals a consistent performance hierarchy among the different architectures:

- **MF-Base:** This model serves as the performance baseline. Its front is positioned in the upper-right, indicating it is the least effective and least efficient option. It is almost entirely dominated by the other architectures.
- **MF+Biases and MF+Mean:** Adding only biases (`MF+Biases`) or only the global mean (`MF+Mean`) both offer improvements over the baseline. The mean provides a more substantial gain, shifting the entire front towards better accuracy and faster convergence. Adding biases primarily accelerates convergence without improving the best achievable RMSE.
- **MF+Mean+Biases (Proposed Model):** This architecture dominates all other variants. Its Pareto front is positioned significantly lower and to the left, proving it can achieve superior RMSE values in far fewer epochs. To illustrate the advantage: our proposed model can match the baseline's best validation RMSE of 0.931 in only 15 epochs, whereas the baseline required 145 epochs. This represents a **9.7x convergence speedup**. Furthermore, this architecture claims the best overall configuration found, achieving a validation RMSE of 0.903 in 97 epochs.

This analysis not only establishes the clear superiority of the full `MF+Mean+Biases` architecture but also provides a principled method for selecting the most representative models for a final evaluation. For this purpose, we selected the best performing set of hyperparameters (based on the lowest achieved validation RMSE) from each of the four fronts. To explicitly quantify the trade-offs, we also included a "fast" configuration of our proposed model, chosen to match the baseline's best accuracy in the minimum number of epochs. The hyperparameters for these five selected models are detailed in Table 6.5 and are subjected to a final evaluation on the held-out test set in the following section.

Table 6.5: Hyperparameters for the models selected for final evaluation.

Model	Type	Dim (d)	LR (η)	Bias LR (η_{bias})	Reg (λ)	Epochs
MF-Base	Best RMSE	6	0.001137	—	0.5341	145
MF+Mean+Biases	Best RMSE	32	0.005590	0.002467	14.11	97
MF+Mean+Biases	Fast	37	0.009100	0.003141	3.634	15
MF+Biases	Best RMSE	6	0.0009946	5.722e-5	1.411e-5	168
MF+Mean	Best RMSE	5	0.007174	—	4.451	123

6.3.3. Evaluation on the Test Set

The models were then subjected to a rigorous final evaluation using our two key metrics: predictive accuracy (RMSE) and ranking quality (nDCG@10). To account for performance variations from the random initialization of the latent factor matrices, each of the five models was trained from scratch and evaluated on the held-out test set 20 times. The performance metrics, averaged over these runs, are visualized in Figure 6.2. Figure 6.3 visualizes the relative improvement of each configuration over the baseline `MF-Base` model.

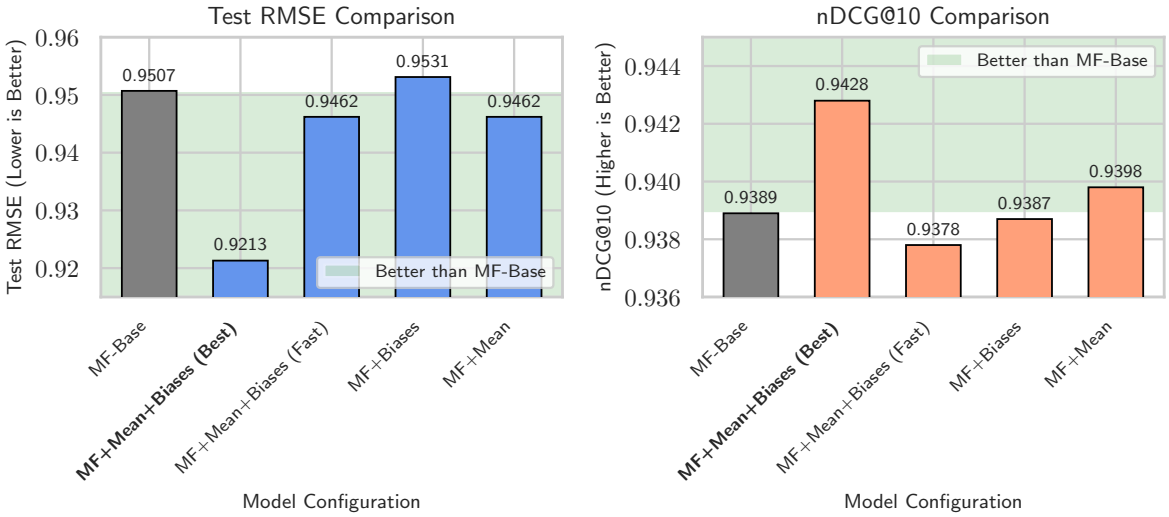

Figure 6.2: Final performance on the test set, averaged over 20 runs.

The results presented in the figures lead to several key conclusions about the models themselves:

- **Dominant Performance of the Proposed Model:** The complete `MF+Mean+Biases` architecture emerges as the top-performing model. It achieves a Test RMSE of 0.9213, a substantial 3.09% relative improvement over the MF-Base model's 0.9507 RMSE (Figure 6.3). Furthermore, it obtains the highest ranking quality with an nDCG@10 score of 0.9428, representing a relative improvement of 0.42%. This demonstrates that the inclusion of both global mean and bias terms is an effective strategy for maximizing predictive accuracy and ranking performance.
- **A Synergistic Effect:** The importance of combining the mean and bias components is illustrated by the intermediate models. Including only the global mean (`MF+Mean`) provides a modest 0.47% relative improvement in RMSE, while adding only biases (`MF+Biases`) did not provide a benefit, instead showing a slight degradation in performance relative to the baseline. However, when used together, they produce the 3.09% RMSE improvement, a gain that far exceeds the sum of their individual effects. This highlights a powerful synergy where the global mean provides a stable predictive base, allowing the bias terms to model user and item tendencies more effectively.
- **Increased Model Capacity and Factor Specialization:** A key insight from the hyperparameter search (Table 6.5) is that the `MF+Mean+Biases` model achieves its optimal performance with a significantly larger latent dimension ($d = 32$) compared to the simpler architectures ($d \approx 5 - 6$). This suggests a specialization of model components. By explicitly modeling the primary effects of

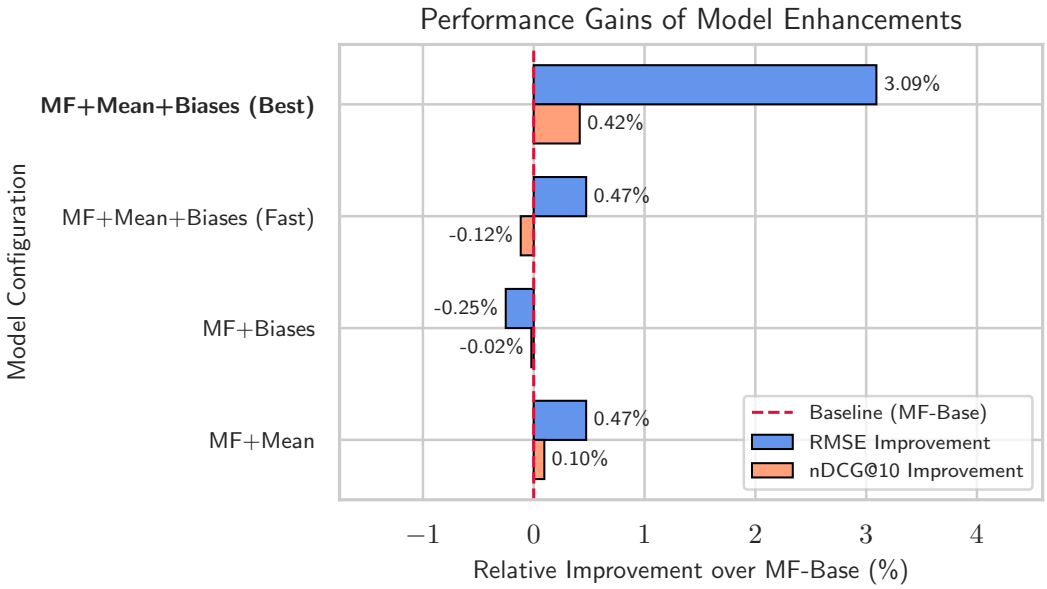

Figure 6.3: Relative improvement in Test RMSE and nDCG@10 for each model configuration compared to the MF-Base baseline. Positive values indicate better performance.

user and item biases, the architecture appears to ”free” the latent factors to focus exclusively on capturing more subtle, higher-order interactions. Simpler models must conflate both tasks within the factors, limiting their expressive power. This specialization would then explain why the full model can effectively leverage a higher-dimensional space, increasing its capacity to learn, and leading to superior predictive accuracy.

- **A Clear Efficiency-Accuracy Trade-off:** The `Fast` configuration highlights the practical value of our approach. It achieves a 0.47% relative RMSE improvement over the baseline, matching the `MF+Mean` model's accuracy, while converging nearly ten times faster (15 epochs vs. 145). However, this efficiency comes at a minor, quantifiable cost to ranking performance, with its nDCG@10 score showing a 0.12% relative decrease compared to the baseline. This trade-off suggests that the hyperparameters optimized for rapid convergence are slightly different from those that perfect the top-10 item ranking, offering a valuable option for scenarios where computational cost is a primary constraint.

6.4. The True Cost of Accuracy

The preceding analyses have established a comprehensive case for our proposed enhancements. We have demonstrated that they are secure by design (section 6.1), introduce a modest and predictable theoretical overhead (section 6.2), and provide significant empirical gains in both accuracy and convergence speed (section 6.3). A simple analysis might conclude here, asserting that these improvements are always worth the minor increase in vector dimensionality.

However, a closer look at our hyperparameter optimization results (Table 6.5) reveals a critical nuance. The true cost of the enhanced architecture is not merely the addition of two bias terms. To reach their peak performance, the enhanced models require a significantly larger latent dimension (d). For instance, the `MF+Mean+Biases` model found its optimum with $d = 32$, giving it a total operational vector dimension of 34, whereas the baseline's optimal was just $d = 6$. This discovery necessitates a more sophisticated cost-benefit analysis than a simple ”+2” dimension comparison. While the architectural advantage is evident, the crucial question becomes: Does this advantage hold up when measured against the true total computational cost?

To answer this, we define a metric that approximates the total computational effort. This metric is proportional to the product of the model's operational dimensionality (d) and the number of iterations

(K) required for convergence:

$$\text{Total Effort} \propto \begin{cases} K \times d & \text{for } \texttt{MF-Base} \\ K \times (d+2) & \text{for } \texttt{MF+Mean+Biases}. \end{cases} \tag{6.5}$$

Figure 6.4 plots this total effort against the Validation RMSE for every Pareto-optimal configuration of the `MF-Base` and `MF+Mean+Biases` models from subsection 6.3.2. This provides a direct comparison of the true computational cost required to achieve a given level of accuracy. The analysis uses validation data, offering a direct view into the efficiency of the model's learning process.

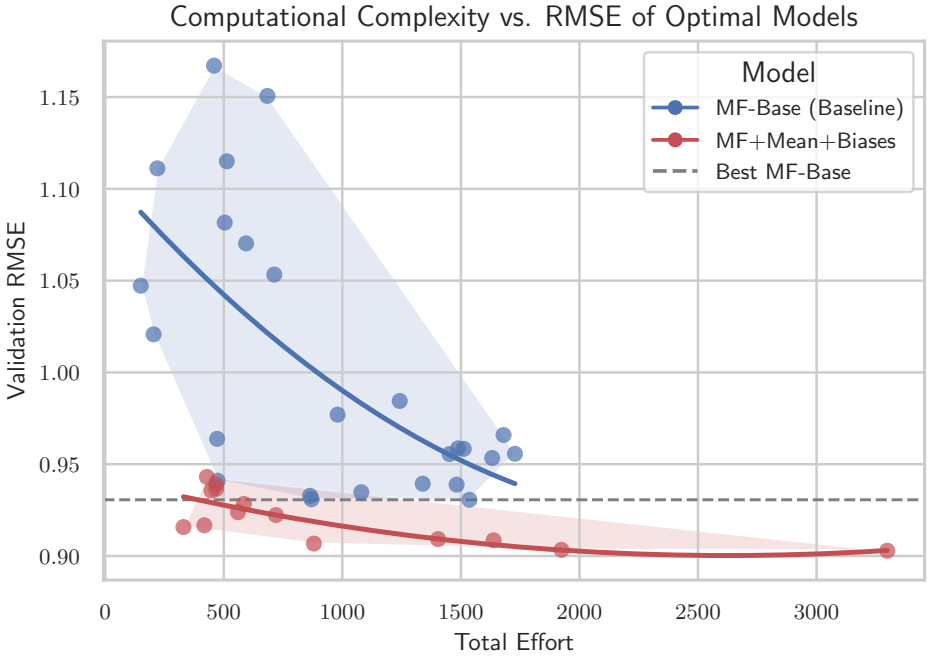

Figure 6.4: Total computational effort vs. Validation RMSE for all Pareto-optimal configurations. Each point represents a non-dominated hyperparameter set. The desirable region (low effort, low error) is the bottom-left. The shaded areas visually represent the performance envelope for each model class, while the dashed line at RMSE=0.931 marks the best performance achieved by any baseline configuration.

The graph highlights a sharp contrast in performance. The baseline models cluster above a clear performance barrier, unable to breach a validation RMSE of 0.931, regardless of the computational effort invested. In contrast, the enhanced models not only break this barrier, achieving higher accuracy but also demonstrate superior performance across the entire computational effort spectrum. This leads to two distinct conclusions depending on the optimization goal:

- **For those seeking maximum performance:** The enhanced model is the only choice. The hyperparameter search could not find a single configuration of the baseline model capable of matching the accuracy of the enhanced architecture. As Figure 6.4 shows, the red dots populate a region of low RMSE that is entirely inaccessible to the blue dots, confirming the existence of a hard performance ceiling for the baseline.
- **For those seeking maximum efficiency:** The enhanced model offers a more cost-effective training process. At any given level of computational effort, the best-achievable RMSE from the enhanced architecture is consistently lower than that of the baseline. This holds true across the entire spectrum, from low-cost configurations (e.g., in the 300-1500 Total Effort range) to high-cost ones, proving the enhanced architecture's superiority when resources are constrained.

The enhanced architecture is not a marginal improvement but a fundamental replacement for the baseline. We have demonstrated that its superiority holds even when accounting for its higher true complexity—both the larger latent dimensions and the two additional bias terms. This dual advantage is especially important in a cryptographic setting, where every operation is inherently expensive;

the model's faster convergence can deliver better results for less computational effort. Furthermore, its higher predictive accuracy translates directly to more relevant and helpful course suggestions for students. Therefore, these architectural enhancements are validated not merely as a theoretical improvement, but as a necessary and practical advancement for building a recommender system that is powerful, efficient, and secure by design.

7

Discussion and Conclusion

7.1. Argument and Interpretation of Findings

A central challenge in educational technology is the tension between data-driven personalization and the non-negotiable need for student privacy. This thesis argues that this is not an inescapable trade-off but a technical challenge that can be met. We demonstrate that cryptographic methods can deliver strong privacy guarantees without a significant penalty to predictive accuracy.

To do so, we designed and evaluated a course recommendation system using Homomorphic Encryption (HE), which guarantees that raw student data is never exposed to any party. Our system builds upon the encrypted matrix factorization protocol of Kim, Kim, Koo, *et al.* [24] by integrating bias terms, a standard technique for improving accuracy in non-private models. Our results show a significant improvement over this privacy-preserving baseline: the proposed model achieved a test RMSE of 0.9213 and an nDCG@10 of 0.9428. This represents a 3.09% improvement in RMSE, while also reaching the baseline's peak performance 9.7 times faster (see Figure 6.3). Furthermore, our approach consistently outperforms the baseline across all levels of computational effort, as demonstrated in Figure 6.4.

7.1.1. The Contribution to Private Recommender Systems

In collaborative filtering, where accuracy gains are often incremental, this improvement represents a significant advance. Our approach makes two key contributions:

1. **Bridging the Performance Gap:** Our model substantially narrows the performance gap between private and non-private systems. By integrating bias terms directly into the encrypted protocol—a crucial enhancement omitted by prior encrypted matrix factorization protocols—we achieve significant performance gains. This integration drives our model's improved test RMSE from 0.951 to 0.921, a substantial improvement in collaborative filtering contexts. To place this result in perspective, Table 7.1 includes several standard non-private benchmarks. It is important to note that the exact performance of these benchmarks can vary based on factors like implementation and hyperparameter optimization. Their purpose here is to demonstrate that our private approach achieves an accuracy level that is firmly on par with, and competitive within the range of, standard non-encrypted methods. While more sophisticated models exist, establishing a competitive benchmark with a foundational method like biased Matrix Factorization is a critical prerequisite for privacy-preserving research in this domain. Its compatibility with homomorphic encryption primitives makes it the ideal starting point. Our results demonstrate that well-designed cryptographic systems can genuinely compete with non-private alternatives.
2. **Establishing a New Benchmark:** This work establishes a new practical benchmark for *privacy-by-design* recommender systems by addressing key limitations of existing approaches. Unlike Differential Privacy methods, our approach avoids inherent accuracy degradation [20]. Compared to encrypted neighborhood-based methods, we achieve superior accuracy on sparse data [81]. Most importantly, we outperform previous encrypted matrix factorization protocols by incorporating the bias terms they omitted [23], [24].

Table 7.1: Summary of Results. Our private, biased MF model shows a significant RMSE improvement over the private baseline and is competitive with standard non-private benchmarks (values from Hug [82]) on the MovieLens-100k dataset.

Category	Algorithm	Test RMSE
Private Models (This Thesis)	Baseline (Kim, Kim, Koo, *et al.* [24])	0.951
	Biased MF (Ours)	**0.921**
Non-Private Benchmarks	Biases Only	0.944
	Biased MF (SVD)	0.934
	Neighborhood-Based (k-NN)	0.931

7.1.2. Uncovering Latent Aptitude

As introduced in subsection 5.3.3, an additional strength of our model is the conceptual shift it enables. By decomposing the prediction into baseline estimates (μ, b_i, b_j) and an interaction term ($\mathbf{u}_i^T \mathbf{v}_j$), the system can answer two distinct and valuable questions:

1. **"In which course am I likely to get a high grade?"** This is answered by the full prediction, which is dominated by general popularity and baseline factors.
2. **"For which course do I have a unique aptitude?"** This is answered by the **latent interaction term** ($\mathbf{u}_i^T \mathbf{v}_j$) alone.

This second component serves as a proxy for the **latent compatibility** between a student and a course, independent of that course's average difficulty or the student's average performance. It isolates the unique, personalized affinity. This transforms the recommender from a simple grade optimizer into a tool for academic discovery, capable of surfacing courses and fields where a student has a hidden talent, guiding them toward more fulfilling academic paths they might otherwise never have considered.

7.2. Limitations and Future Directions

While the findings are promising, they were established under controlled conditions. A critical appraisal of the study's limitations is essential for contextualizing the results and defining a clear trajectory for future inquiry. These limitations can be divided into methodological choices and architectural gaps in the protocol.

7.2.1. Methodological Limitations

The first limitation is the use of the `MovieLens-100k` proxy dataset. While a standard benchmark, the dynamics of movie ratings may not perfectly mirror those of student course selection and performance. A definitive validation of our model requires testing on a real-world educational dataset to confirm its effectiveness in the target domain.

The second limitation is the evaluation in plaintext. While this methodology is valid for demonstrating the *algorithmic superiority* of our approach, it does not capture the full picture. The numerical results do not carry over perfectly to the encrypted domain due to the use of fixed-point arithmetic, which introduces small precision errors. However, as shown by Kim, Kim, Koo, *et al.* [24], these errors can be made negligible with appropriate parameter selection. A full-stack deployment is still required to benchmark the system's *practicality*, accounting for the considerable engineering challenges of cryptographic parameter optimization, computational overhead, and network latency.

7.2.2. Architectural Limitations

We have also identified two architectural limitations, partially inherited from the foundational protocol, that persist in our design and should be addressed in future work:

- **Lack of Gradient Normalization:** In Batch Gradient Descent, it is standard practice to normalize the summed gradient by the total number of ratings to ensure the effective learning rate remains stable as the dataset grows [83]. The current protocol does not perform this normalization, a feature that is important for any adaptive, real-world system.
- **Single Learning Rate:** Our plaintext hyperparameter optimization (Table 6.5) found that the best-

performing models used different, and often smaller, learning rates for the bias terms compared to the latent factors. However, the cryptographic protocol in its current form supports only a single learning rate for all parameters. This implies that a direct implementation of the cryptographic protocol would likely not achieve the optimal test RMSE of 0.9213 reported in our evaluation. A next step is to modify the protocol to support separate learning rates, allowing the privacy-preserving model to realize its full, demonstrated potential.

7.2.3. Future Research

Addressing these limitations points toward a clear path for future research. The immediate priority is to implement the architectural improvements (gradient normalization and separate learning rates) and validate the complete system in a real-world educational setting. Such a study should be designed not merely to confirm performance but to test the novel hypotheses enabled by our model:

1. Does the incorporation of high school academic records provide a statistically significant improvement in mitigating the cold-start problem for first-year students?
2. Is there a demonstrable correlation between a high latent affinity score ($\mathbf{u}_i^T \mathbf{v}_j$) and qualitative measures of student success, such as course engagement, long-term interest, or overall academic satisfaction?

These real-world validation studies represent the immediate future for this line of educational technology research. In parallel, the broader field of privacy-preserving machine learning continues to address the long-term ”grand challenge”: closing the performance gap with non-private, state-of-the-art neural architectures like Graph Neural Networks or Transformers [84]. Adapting these complex models to the HE framework remains a significant challenge due to the high cost of encrypted non-linear operations [85], but it is a path that will be informed by the validated success of fundamental models like the one presented in this thesis.

7.3. Conclusion

This thesis began with the tension between personalization and privacy in education and concludes with a demonstration that this tension can be productively resolved. By developing and validating an enhanced, privacy-preserving matrix factorization model, this work has shown that it is possible to build a recommendation system that is both effective and secure by design. The core contribution is not just an algorithm, but a proof of concept: that we can engineer systems that reflect our values. Moving forward, the goal of this research field should not be limited to creating tools that are merely intelligent, but to building a new generation of educational technology that is demonstrably respectful of student autonomy, worthy of institutional trust, and fundamentally committed to safeguarding the privacy of those it is designed to serve.

References

[1] T. Hussey and P. Smith, "Transitions in higher education," *Innovations in Education and Teaching International*, vol. 47, no. 2, pp. 155–164, May 2010, ISSN: 1470-3297, 1470-3300. DOI: 10.1080/14703291003718893. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/14703291003718893 (visited on 06/16/2025).

[2] K. Baird, "An inquiry into withdrawal from college a study conducted at trinity college dublin." [Online]. Available: https://www.tcd.ie/media/tcd/student-counselling/pdfs/Report_for_withdrawal_study.pdf (visited on 06/27/2025).

[3] C. Learning. "4 challenges derailing advisors and strategies to fix them," Civitas Learning. (Sep. 13, 2022), [Online]. Available: https://www.civitaslearning.com/blog/4-challenges-derailing-academic-advisor-effectiveness-and-how-to-fix-them/ (visited on 06/16/2025).

[4] J. Komljenovic, K. Birch, S. Sellar, *et al.*, "Digitalised higher education: Key developments, questions, and concerns," *Discourse: Studies in the Cultural Politics of Education*, vol. 46, no. 2, pp. 276–292, Mar. 4, 2025, ISSN: 0159-6306, 1469-3739. DOI: 10.1080/01596306.2024.2408397. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01596306.2024.2408397 (visited on 06/16/2025).

[5] M. Klose, V. Desai, and Y. Song, "EDM and privacy: Ethics and legalities of data collection, usage, and storage," presented at the 13th International Conference on Educational Data Mining, 2020.

[6] "EDUCAUSE QuickPoll results: Third-party risk management practices in higher education," EDUCAUSE Review. (), [Online]. Available: https://er.educause.edu/articles/2024/8/educause-quickpoll-results-third-party-risk-management-practices-in-higher-education (visited on 06/16/2025).

[7] "FTC says ed tech provider edmodo unlawfully used children's personal information for advertising and outsourced compliance to school districts," Federal Trade Commission. (May 22, 2023), [Online]. Available: https://www.ftc.gov/news-events/news/press-releases/2023/05/ftc-says-ed-tech-provider-edmodo-unlawfully-used-childrens-personal-information-advertising (visited on 06/25/2025).

[8] W. News. "Data breach exposed personal info of nearly 6,000 montgomery county student accounts," WTOP News. (Dec. 3, 2019), [Online]. Available: https://wtop.com/montgomery-county/2019/12/data-breach-exposed-personal-info-of-nearly-6000-montgomery-county-student-accounts/ (visited on 06/27/2025).

[9] A. Yan. "Chinese court tries fraud suspects after victim dies of heart attack," CGTN News. (Jun. 27, 2017), [Online]. Available: https://news.cgtn.com/news/3d49444d3551444e/share_p.html (visited on 06/27/2025).

[10] The Sedona Conference, "Commentary on data privacy and security issues in mergers & acquisitions practice," *The Sedona Conference Journal*, J. 233, vol. 20, J. 233 2019. [Online]. Available: https://thesedonaconference.org/sites/default/files/publications/Data%20Privacy%20and%20Security%20Issues%20in%20Mergers%20and%20Acquisitions%20%282019%29.pdf (visited on 06/16/2025).

[11] S. Gurses, C. Troncoso, and C. Diaz, "Engineering privacy by design,"

[12] D. B. Guruge, R. Kadel, and S. J. Halder, "The state of the art in methodologies of course recommender systems—a review of recent research," *Data*, vol. 6, no. 2, p. 18, Feb. 11, 2021, ISSN: 2306-5729. DOI: 10.3390/data6020018. [Online]. Available: https://www.mdpi.com/2306-5729/6/2/18 (visited on 08/29/2024).

[13] M. Charles and K. Bradley, “Indulging our gendered selves? sex segregation by field of study in 44 countries,” *American Journal of Sociology*, vol. 114, no. 4, pp. 924–976, Jan. 2009, ISSN: 0002-9602, 1537-5390. DOI: 10.1086/595942. [Online]. Available: http://www.journals.uchicago.edu/doi/10.1086/595942 (visited on 06/16/2025).

[14] A. P. Carnevale and J. Strohl, “Separate & unequal, how higher education reinforces the intergenerational reproduction of white racial privilege,” 2009. [Online]. Available: https://cew.georgetown.edu/wp-content/uploads/SeparateUnequal.FR_.pdf (visited on 06/27/2025).

[15] K. Danilowicz-Gösele, K. Lerche, J. Meya, and R. Schwager, “Determinants of students’ success at university,” *Education Economics*, vol. 25, no. 5, pp. 513–532, Sep. 3, 2017, ISSN: 0964-5292, 1469-5782. DOI: 10.1080/09645292.2017.1305329. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09645292.2017.1305329 (visited on 05/02/2024).

[16] H.-L. Thanh-Nhan, H.-H. Nguyen, and N. Thai-Nghe, “Methods for building course recommendation systems,” in *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, Oct. 2016, pp. 163–168. DOI: 10.1109/KSE.2016.7758047. [Online]. Available: https://ieeexplore.ieee.org/document/7758047/?arnumber=7758047 (visited on 07/26/2024).

[17] R. Bell and Y. Koren, “Advances in collaborative filtering,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris France: ACM, Jun. 28, 2009, pp. 447–456, ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557072. [Online]. Available: https://dl.acm.org/doi/10.1145/1557019.1557072 (visited on 08/20/2024).

[18] A. Narayanan and V. Shmatikov, “Robust de-anonymization of large sparse datasets,” in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, ISSN: 2375-1207, May 2008, pp. 111–125. DOI: 10.1109/SP.2008.33. [Online]. Available: https://ieeexplore.ieee.org/document/4531148/?arnumber=4531148 (visited on 03/24/2025).

[19] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3, pp. 211–407, 2013, ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000042. [Online]. Available: http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042 (visited on 03/28/2025).

[20] F. McSherry and I. Mironov, “Differentially private recommender systems: Building privacy into the netflix prize contenders,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris France: ACM, Jun. 28, 2009, pp. 627–636, ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557090. [Online]. Available: https://dl.acm.org/doi/10.1145/1557019.1557090 (visited on 08/11/2025).

[21] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, “Communication-efficient learning of deep networks from decentralized data,” 2017.

[22] Z. Li, B. Ding, C. Zhang, N. Li, and J. Zhou, “Federated matrix factorization with privacy guarantee,” *Proceedings of the VLDB Endowment*, vol. 15, no. 4, pp. 900–913, Dec. 2021, ISSN: 2150-8097. DOI: 10.14778/3503585.3503598. [Online]. Available: https://dl.acm.org/doi/10.14778/3503585.3503598 (visited on 03/26/2025).

[23] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, “Privacy-preserving matrix factorization,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS ’13*, Berlin, Germany: ACM Press, 2013, pp. 801–812, ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516751. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508859.2516751 (visited on 09/09/2024).

[24] S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, and J. Shin, “Efficient privacy-preserving matrix factorization via fully homomorphic encryption: Extended abstract,” in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, Xi’an China: ACM, May 30, 2016, pp. 617–628, ISBN: 978-1-4503-4233-9. DOI: 10.1145/2897845.2897875. [Online]. Available: https://dl.acm.org/doi/10.1145/2897845.2897875 (visited on 12/03/2024).

[25] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, Conference Name: Computer, ISSN: 1558-0814. DOI: 10.1109/MC.2009.263. [Online]. Available: https://ieeexplore.ieee.org/document/5197422/?arnumber=5197422&tag=1 (visited on 11/18/2024).

[26] N. Kaaniche, M. Laurent, and S. Belguith, “Privacy enhancing technologies for solving the privacy-personalization paradox: Taxonomy and survey,” *Journal of Network and Computer Applications*, vol. 171, p. 102 807, Dec. 2020, ISSN: 10848045. DOI: 10.1016/j.jnca.2020.102807. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804520302794 (visited on 08/06/2025).

[27] L. Sweeney, “Simple demographics often identify people uniquely,” Carnegie Mellon University, Pittsburgh, Data Privacy Working Paper 3, 2000. [Online]. Available: https://dataprivacylab.org/projects/identifiability/paper1.pdf (visited on 06/08/2025).

[28] L. Sweeney, “K-ANONYMITY: A MODEL FOR PROTECTING PRIVACY,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, Oct. 2002, ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S0218488502001648. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0218488502001648 (visited on 08/06/2025).

[29] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “*L* -diversity: Privacy beyond *k* -anonymity,” *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, p. 3, Mar. 2007, ISSN: 1556-4681, 1556-472X. DOI: 10.1145/1217299.1217302. [Online]. Available: https://dl.acm.org/doi/10.1145/1217299.1217302 (visited on 08/06/2025).

[30] N. Li, T. Li, and S. Venkatasubramanian, “T-closeness: Privacy beyond k-anonymity and l-diversity,” presented at the 2007 IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey: IEEE (Institute of Electrical and Electronics Engineers), Apr. 2007, pp. 106–115.

[31] W. Diffie and M. E. Hellman, “Multiuser cryptographic techniques,” in *Proceedings of the June 7-10, 1976, national computer conference and exposition on - AFIPS ’76*, New York, New York: ACM Press, 1976, p. 109. DOI: 10.1145/1499799.1499815. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1499799.1499815 (visited on 08/06/2025).

[32] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” *ACM Computing Surveys*, vol. 51, no. 4, 79:1–79:35, Jul. 25, 2018, ISSN: 0360-0300. DOI: 10.1145/3214303. [Online]. Available: https://dl.acm.org/doi/10.1145/3214303 (visited on 03/21/2024).

[33] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Advances in Cryptology — EUROCRYPT ’99*, J. Stern, Ed., vol. 1592, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238, ISBN: 978-3-540-65889-4. DOI: 10.1007/3-540-48910-X_16. [Online]. Available: http://link.springer.com/10.1007/3-540-48910-X_16 (visited on 04/25/2025).

[34] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, Bethesda MD USA: ACM, May 31, 2009, pp. 169–178, ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536440. [Online]. Available: https://dl.acm.org/doi/10.1145/1536414.1536440 (visited on 08/06/2025).

[35] V. Lyubashevsky, C. Peikert, and O. Regev, “Ideal lattices and learning with errors over rings,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–20. DOI: 10.1007/978-3-642-14518-6_3.

[36] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical GapSVP,” in *Advances in Cryptology – CRYPTO 2012*, vol. 7417, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886, ISBN: 978-3-642-32008-8 978-3-642-32009-5. DOI: 10.1007/978-3-642-32009-5_50. [Online]. Available: http://link.springer.com/10.1007/978-3-642-32009-5_50 (visited on 08/06/2025).

[37] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, Jul. 2014, ISSN: 1942-3454, 1942-3462. DOI: 10.1145/2633600. [Online]. Available: https://dl.acm.org/doi/10.1145/2633600 (visited on 08/06/2025).

[38] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” in *International Workshop on Post-Quantum Cryptography*, Springer, 2012, pp. 245–262.

[39] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology – ASIACRYPT 2017*, vol. 10624, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, pp. 409–437, ISBN: 978-3-319-70693-1 978-3-319-70694-8. DOI: 10.1007/978-3-319-70694-8_15. [Online]. Available: https://link.springer.com/10.1007/978-3-319-70694-8_15 (visited on 08/06/2025).

[40] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “TFHE: Fast fully homomorphic encryption over the torus,” *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, Jan. 2020, ISSN: 0933-2790, 1432-1378. DOI: 10.1007/s00145-019-09319-x. [Online]. Available: http://link.springer.com/10.1007/s00145-019-09319-x (visited on 08/06/2025).

[41] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, “”you might also like:” privacy risks of collaborative filtering,” in *2011 IEEE Symposium on Security and Privacy*, Oakland, CA, USA: IEEE, May 2011, pp. 231–246, ISBN: 978-1-4577-0147-4. DOI: 10.1109/SP.2011.40. [Online]. Available: http://ieeexplore.ieee.org/document/5958032/ (visited on 08/11/2025).

[42] Y. S. Resheff, Y. Elazar, M. Shahar, and O. S. Shalom, *Privacy and fairness in recommender systems via adversarial training of user representations*, Dec. 18, 2018. DOI: 10.48550/arXiv.1807.03521. arXiv: 1807.03521[cs]. [Online]. Available: http://arxiv.org/abs/1807.03521 (visited on 08/11/2025).

[43] H. Polat and Wenliang Du, “Privacy-preserving collaborative filtering using randomized perturbation techniques,” in *Third IEEE International Conference on Data Mining*, Melbourne, FL, USA: IEEE Comput. Soc, 2003, pp. 625–628, ISBN: 978-0-7695-1978-4. DOI: 10.1109/ICDM.2003.1250993. [Online]. Available: http://ieeexplore.ieee.org/document/1250993/ (visited on 08/11/2025).

[44] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” *Theory of Cryptography Conference (TCC 2006)*, vol. 3876, pp. 1–19, 2006.

[45] H. Shin, S. Kim, J. Shin, and X. Xiao, “Privacy enhanced matrix factorization for recommendation with local differential privacy,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1770–1782, Sep. 1, 2018, ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2018.2805356. [Online]. Available: https://ieeexplore.ieee.org/document/8290673/ (visited on 08/11/2025).

[46] D. Chai, L. Wang, K. Chen, and Q. Yang, “Secure federated matrix factorization,” *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 11–20, Sep. 2021, Conference Name: IEEE Intelligent Systems, ISSN: 1941-1294. DOI: 10.1109/MIS.2020.3014880. [Online]. Available: https://ieeexplore.ieee.org/document/9162459/?arnumber=9162459 (visited on 10/07/2024).

[47] M. Ammad-ud-din, E. Ivannikova, S. A. Khan, *et al.*, *Federated collaborative filtering for privacy-preserving personalized recommendation system*, Jan. 29, 2019. arXiv: 1901.09888[cs]. [Online]. Available: http://arxiv.org/abs/1901.09888 (visited on 10/15/2024).

[48] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” presented at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 2019, pp. 14 747–14 756.

[49] B. Zhao, K. R. Mopuri, and H. Bilen, *iDLG: Improved deep leakage from gradients*, Jan. 8, 2020. DOI: 10.48550/arXiv.2001.02610. arXiv: 2001.02610[cs]. [Online]. Available: http://arxiv.org/abs/2001.02610 (visited on 08/11/2025).

[50] W. Wei, L. Liu, M. Loper, *et al.*, *A framework for evaluating gradient leakage attacks in federated learning*, Apr. 23, 2020. DOI: 10.48550/arXiv.2004.10397. arXiv: 2004.10397[cs]. [Online]. Available: http://arxiv.org/abs/2004.10397 (visited on 08/11/2025).

[51] Z. Xu, C. Chu, and S. Song, “An effective federated recommendation framework with differential privacy,” *Electronics*, vol. 13, no. 8, p. 1589, Apr. 22, 2024, ISSN: 2079-9292. DOI: 10.3390/electronics13081589. [Online]. Available: https://www.mdpi.com/2079-9292/13/8/1589 (visited on 08/11/2025).

[52] J. Canny, “Collaborative filtering with privacy,” in *Proceedings 2002 IEEE Symposium on Security and Privacy*, Berkeley, CA, USA: IEEE Comput. Soc, 2002, pp. 45–57, ISBN: 978-0-7695-1543-4. DOI: 10.1109/SECPRI.2002.1004361. [Online]. Available: http://ieeexplore.ieee.org/document/1004361/ (visited on 08/19/2024).

[53] J. Canny, “Collaborative filtering with privacy via factor analysis,” 2002.

[54] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, “Privacy enhanced recommender system,” 2010.

[55] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, “Efficiently computing private recommendations,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, ISSN: 2379-190X, May 2011, pp. 5864–5867. DOI: 10.1109/ICASSP.2011.5947695. [Online]. Available: https://ieeexplore.ieee.org/document/5947695/ (visited on 08/11/2025).

[56] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, “Generating private recommendations efficiently using homomorphic encryption and data packing,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, Jun. 2012, ISSN: 1556-6021. DOI: 10.1109/TIFS.2012.2190726. [Online]. Available: https://ieeexplore.ieee.org/document/6168832/ (visited on 08/11/2025).

[57] M. Scheffel, H. Drachsler, S. Stoyanov, and M. Specht, “Quality indicators for learning analytics,” *Educational Technology & Society*, no. 17, pp. 117–132, Jan. 10, 2014.

[58] A. Hellas, P. Ihantola, A. Petersen, *et al.*, “Predicting academic performance: A systematic literature review,” in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, Larnaca Cyprus: ACM, Jul. 2, 2018, pp. 175–199, ISBN: 978-1-4503-6223-8. DOI: 10.1145/3293881.3295783. [Online]. Available: https://dl.acm.org/doi/10.1145/3293881.3295783 (visited on 08/12/2025).

[59] N. Mduma, K. Kalegele, and D. Machuve, “A survey of machine learning approaches and techniques for student dropout prediction,” *Data Science Journal*, vol. 18, p. 14, Apr. 17, 2019, ISSN: 1683-1470. DOI: 10.5334/dsj-2019-014. [Online]. Available: http://datascience.codata.org/articles/10.5334/dsj-2019-014/ (visited on 08/12/2025).

[60] P. Kaur, A. Kaur, and R. Kaur, “A systematic review about prediction of academic behavior through data mining techniques,” *Journal of Computational and Theoretical Nanoscience*, vol. 17, no. 11, pp. 5162–5166, Nov. 1, 2020, ISSN: 1546-1955. DOI: 10.1166/jctn.2020.9358. [Online]. Available: https://www.ingentaconnect.com/content/10.1166/jctn.2020.9358 (visited on 08/12/2025).

[61] C. Y. Chuang, S. D. Craig, and J. Femiani, “Detecting probable cheating during online assessments based on time delay and head pose,” *Higher Education Research & Development*, vol. 36, no. 6, pp. 1123–1137, Sep. 19, 2017, ISSN: 0729-4360, 1469-8366. DOI: 10.1080/07294360.2017.1303456. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/07294360.2017.1303456 (visited on 08/12/2025).

[62] L. C. O. Tiong and H. J. Lee, *E-cheating prevention measures: Detection of cheating at online examinations using deep learning approach – a case study*, Jan. 25, 2021. DOI: 10.48550/arXiv.2101.09841. arXiv: 2101.09841[cs]. [Online]. Available: http://arxiv.org/abs/2101.09841 (visited on 08/12/2025).

[63] P. Muncaster. “Almost 900 US schools breached via MOVEit,” Infosecurity Magazine. (Sep. 25, 2023), [Online]. Available: https://www.infosecurity-magazine.com/news/us-900-schools-breached-moveit/ (visited on 08/12/2025).

[64] J. R. Reidenberg and F. Schaub, “Achieving big data privacy in education,” *Theory and Research in Education*, vol. 16, no. 3, pp. 263–279, Nov. 2018, ISSN: 1477-8785, 1741-3192. DOI: 10.1177/1477878518805308. [Online]. Available: https://journals.sagepub.com/doi/10.1177/1477878518805308 (visited on 08/12/2025).

[65] E. Yacobson, S. Hershkovitz, O. Fuhrman, and G. Alexandron, “De-identification is not enough to guarantee student privacy: De- anonymizing personal information from basic logs,” Jul. 2020.

[66] G. Chen, D. Davis, J. Lin, C. Hauff, and G.-J. Houben, “Beyond the MOOC platform: Gaining insights about learners from the social web,” in *Proceedings of the 8th ACM Conference on Web Science*, Hannover Germany: ACM, May 22, 2016, pp. 15–24, ISBN: 978-1-4503-4208-7. DOI: 10.1145/2908131.2908145. [Online]. Available: https://dl.acm.org/doi/10.1145/2908131.2908145 (visited on 07/01/2024).

[67] K. E. Arnold and N. Sclater, “Student perceptions of their privacy in leaning analytics applications,” in *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, Vancouver British Columbia Canada: ACM, Mar. 13, 2017, pp. 66–69, ISBN: 978-1-4503-4870-6. DOI: 10.1145/3027385.3027392. [Online]. Available: https://dl.acm.org/doi/10.1145/3027385.3027392 (visited on 08/12/2025).

[68] S. Slade and P. Prinsloo, “Learning analytics: Ethical issues and dilemmas,” *American Behavioral Scientist*, vol. 57, no. 10, pp. 1510–1529, Oct. 2013, ISSN: 0002-7642, 1552-3381. DOI: 10.1177/0002764213479366. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0002764213479366 (visited on 08/12/2025).

[69] A. Pardo and G. Siemens, “Ethical and privacy principles for learning analytics,” *British Journal of Educational Technology*, vol. 45, no. 3, pp. 438–450, May 2014, ISSN: 0007-1013, 1467-8535. DOI: 10.1111/bjet.12152. [Online]. Available: https://bera-journals.onlinelibrary.wiley.com/doi/10.1111/bjet.12152 (visited on 08/12/2025).

[70] S. K. Banihashem, K. Aliabadi, S. Pourroostaei Ardakani, A. Delaver, and M. Nili Ahmadabadi, “Learning analytics: A systematic literature review,” *Interdisciplinary Journal of Virtual Learning in Medical Sciences*, vol. 9, no. 2, Jun. 12, 2018, ISSN: 2476-7263, 2476-7271. DOI: 10.5812/ijvlms.63024. [Online]. Available: https://ijvlms.sums.ac.ir/article_44834.html (visited on 08/12/2025).

[71] L. Paquette, Z. Li, R. Baker, J. Ocumpaugh, and A. Andres, “Who’s learning? using demographics in EDM research,” *Journal of Educational Data Mining*, vol. 12, no. 3, 2020.

[72] H. Drachsler and W. Greller, “Privacy and analytics: It’s a DELICATE issue a checklist for trusted learning analytics,” in *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK ’16*, Edinburgh, United Kingdom: ACM Press, 2016, pp. 89–98, ISBN: 978-1-4503-4190-5. DOI: 10.1145/2883851.2883893. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2883851.2883893 (visited on 08/12/2025).

[73] R. Hasan and M. Fritz, “Understanding utility and privacy of demographic data in education technology by causal analysis and adversarial-censoring,” *Proceedings on Privacy Enhancing Technologies*, vol. 2022, no. 2, pp. 245–262, Apr. 1, 2022, ISSN: 2299-0984. DOI: 10.2478/popets-2022-0044. [Online]. Available: https://petsymposium.org/popets/2022/popets-2022-0044.php (visited on 06/27/2024).

[74] N. Bosch, R. W. Crues, and N. Shaik, ““hello, [REDACTED]”: Protecting student privacy in analyses of online discussion forums,” presented at the 13th International Conference on Educational Data Mining, 2020.

[75] S. Gurses, C. Troncoso, and C. Diaz, “Engineering privacy by design,” 2011.

[76] M. Bellare, V. T. Hoang, and P. Rogaway, “Foundations of garbled circuits,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, Raleigh North Carolina USA: ACM, Oct. 16, 2012, pp. 784–796, ISBN: 978-1-4503-1651-4. DOI: 10.1145/2382196.2382279. [Online]. Available: https://dl.acm.org/doi/10.1145/2382196.2382279 (visited on 04/23/2025).

[77] A. Cakmak, “Predicting student success in courses via collaborative filtering,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 1, no. 5, pp. 10–17, Mar. 30, 2017, ISSN: 2147-6799. DOI: 10.18201/ijisae.2017526690. [Online]. Available: https://www.ijisae.org/IJISAE/article/view/484 (visited on 07/26/2024).

[78] M. Mansoury, H. Abdollahpouri, M. Pechenizkiy, B. Mobasher, and R. Burke, “Feedback loop and bias amplification in recommender systems,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Virtual Event Ireland: ACM, Oct. 19, 2020, pp. 2145–2148, ISBN: 978-1-4503-6859-9. DOI: 10.1145/3340531.3412152. [Online]. Available: https://dl.acm.org/doi/10.1145/3340531.3412152.

[79] F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, Jan. 7, 2016, Publisher: Association for Computing Machinery (ACM), ISSN: 2160-6455, 2160-6463. DOI: 10.1145/2827872. [Online]. Available: https://dl.acm.org/doi/10.1145/2827872 (visited on 07/10/2025).

[80] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, Jul. 25, 2019, pp. 2623–2631, ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330701. [Online]. Available: https://dl.acm.org/doi/10.1145/3292500.3330701 (visited on 08/14/2025).

[81] P. H. Aditya, I. Budi, and Q. Munajat, “A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for e-commerce in indonesia: A case study PT x,” in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Malang, Indonesia: IEEE, Oct. 2016, pp. 303–308, ISBN: 978-1-5090-4629-4. DOI: 10.1109/ICACSIS.2016.7872755. [Online]. Available: http://ieeexplore.ieee.org/document/7872755/.

[82] N. Hug. “Surpriselib,” Surprise. (2024), [Online]. Available: https://surpriselib.com/ (visited on 08/13/2025).

[83] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org (visited on 08/13/2025).

[84] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–38, Jan. 31, 2020, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3285029. [Online]. Available: https://dl.acm.org/doi/10.1145/3285029 (visited on 08/13/2025).

[85] B. Pulido-Gaytan, A. Tchernykh, J. M. Cortés-Mendoza, *et al.*, “Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities,” *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1666–1691, May 2021, ISSN: 1936-6442, 1936-6450. DOI: 10.1007/s12083-021-01076-8. [Online]. Available: https://link.springer.com/10.1007/s12083-021-01076-8 (visited on 08/13/2025).