Privacy analysis of *X-Rec*: algorithms and privacy guarantees

1 Introduction

1.1 *X-Rec*'s setting

Recall that three main parties are involved in *X-Rec*: (1) a user via a proxy, (2) an untrusted recommender, and (3) an untrusted service provider. The trusted party, however, is only involved in the one-time setup of *X-Rec*.[1]

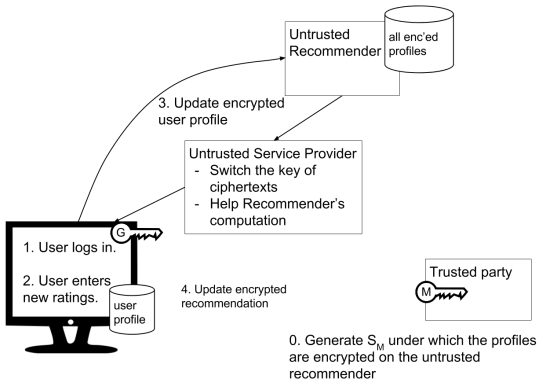

Figure 1: The *X-Rec* architecture

Figure 1 summarizes the interaction between these parties. The trusted party generates the key S_M, distributes useful information (including keys S_G, S_M and key-switching matrices) to the other parties and quit the system. Users input their profiles (encrypted under key S_G) and the recommender outputs predicted ratings (encrypted under S_M) to users. The service provider helps to switch the encryption of predicted ratings from S_M to S_G and helps computation over ciphertexts (which are encryption under key S_M) on the recommender as well. With S_G, users decrypt those ratings and order them locally to have recommendations. The detailed algorithms are described in Section 4.

[1] Hereafter, we stick to the notations above (to mention these parties as the proxy, the recommender, the service provider and the trusted party). In *X-Rec*, the proxy is called the *x-client* (and is considered trusted by the user); the recommender is called the *x-server*; the service provider is called the TP; the trusted party, however, is called the TTP (to distinguish from the service provider).

1.2 Threats

X-Rec addresses mainly two threats. The first threat is a curious recommender who tries to learn private data (e.g. gender, ethnicity, home location) by snooping on user profiles. The second threat is a curious user who tries to learn private data by querying recommendations.

1.3 Privacy guarantees

In *X-Rec*, neither the recommender nor the service provider (computationally bounded) is able to learn anything about user profiles. We summarize these as the *system-level* privacy. As a result, with *X-Rec*, theft of private data, or even sale of private data does not leak information about a single user since all data is encrypted. In *X-Rec*, profiles are encrypted and hence curious database administrators (DBAs) are tolerated. We can thus even outsource the database (and the tasks of the recommender as a whole) to a cloud computing environment.

The recommendation algorithm of *X-Rec* satisfies, in addition, the *user-level* privacy. Roughly speaking, what a user v learns from the output is indistinguishable between two cases: whether another user u participates in *X-Rec* or not. Here the indistinguishability is measured by (ϵ, δ)-indistinguishability [4]; and $\epsilon = \ln \frac{1}{1-F}, \delta = F$ where $F \in (0, 1]$ is a parameter of *X-Rec*.

Thus *X-Rec* ensures both system-level privacy and user-level privacy. To our knowledge, *X-Rec* is the first recommender system to provide both guarantees.

1.4 Organization

The rest of this report is organized as follows. Section 2.2 introduces the adversary model and definitions of system-level privacy and user-level privacy. Section 3 introduces homomorphic encryption and describes our encryption scheme *X-HE* implemented in *X-Rec*. Section 4 describes the important algorithms in *X-Rec* focusing on their privacy perspective. Section 5 proves that *X-Rec* achieves both system-level privacy and user-level privacy. Section 6 discusses the impact of alternative adversary models on *X-Rec* to better understand the privacy guarantee of *X-Rec*.

2 Model and Definitions

2.1 Adversary model

X-Rec provides the privacy guarantees based on the following adversary model: (1) the recommender and the service provider do not collude; (2) all corrupted parties are honest-but-curious. (3) all parties trust the trusted party which is only involved at the one-time setup; and (4) no party can be masqueraded and no message between any two parties can be eavesdropped, injected or modified; (5) corrupted parties have no auxiliary knowledge of other users' profiles (who are users other than the corrupted parties themselves).

The last assumption models the scenario where an ordinary user might be honest-but-curious but has no previous knowledge of any other user. Another remark of the fifth assumption is that as shown later, the recommender does not see any user profile in clear and cannot be the major source of auxiliary knowledge of other users' profiles (as claimed by some attacks, e.g. [22], against recommender systems).

The third and the fourth assumptions are already in use these days. For example, communication between any two parties could be via Transport Layer Security (TLS) with the help of Certificate Authority (CA) so that no party would be masqueraded and no message would be tampered. CA can also play the role of the trusted party.

The first two assumptions are also reasonable as we argue below.

For the first assumption, the service provider is also untrusted but contractually bound to do no collusion with the recommender. Privad [13], a practical private online advertising system, assumed the same model and called it "the dealer mechanism"; the dealer mechanism "is in use today and is approved for instance by the European privacy certification organization Europrise" [13]. Privacy-preserving recommender systems such as Alambic [1], pTwitterRec [18] also assumed no collusion between servers.

For the second assumption, both the recommender and the service provider are willing to follow the system specification. The incentive for the former is to accomplish its promise for recommendation services and to increase revenues, while that for the latter is the consciousness for privacy and should also be contractually bound. Elmisery et al.'s collaborative private framework for an IPTV recommender service [9], Erkin et al.'s and Jeckmans et al.'s private recommenders [10, 14] assumed the same honest-but-curious adversary model.

2.2 Privacy Definitions

2.3 System-level privacy

At the level of the system, we examine the functionality expected from *X-Rec*, which we define as the *ideal* system below.

Definition 1 (System-level privacy [5, 10, 14])**.** We say that a multiparty system α ensures *system-level* privacy if for every *computationally-bounded* algorithm $\mathcal{A}$ that *corrupts* a set of parties when running α, there exists a computationally-bounded algorithm $\mathcal{A}^*$ that corrupts the same set of parties in the ideal system (e.g., Definition 2 for recommendation) such that for every possible input, the joint output of all parties and $\mathcal{A}$ in α and the joint output of all parties and $\mathcal{A}^*$ in the ideal system are computationally indistinguishable.

This notion of system-level privacy is classical in the literature and called "the security of multiparty computation". It was formalized by [5] and has been used by Erkin et al.'s and Jeckmans et al.'s private recommenders [10, 14].

In the context of *X-Rec*, we consider only computationally-bounded adversaries, i.e., the curious party (or parties) in *X-Rec* can only run an

algorithm that is a polynomial of the security parameter of *X-Rec*. In addition, when we say $\mathcal{A}$ corrupts a set of parties, we mean that the set of parties are honest-but-curious in *X-Rec*, and moreover, they behave as defined previously in the adversary model.

Next, we define the ideal recommendation system in the setting of *X-Rec* in Definition 2. Clearly, if *X-Rec* satisfies Definition 2 and Definition 1, then the system-level privacy of *X-Rec* is reduced to the semantic security of *X-HE*. We note that F in Definition 2 can be considered as universally trusted: by Definition 1, $\mathcal{A}^*$ in the ideal system does not corrupt F. Thus users' inputs are only exposed to the universally trusted F, which makes the system ideal.

Definition 2 (Ideal recommendation)**.** The ideal recommendation system is composed of users, the recommender, the service provider, and an additional party: F. Users, the recommender and the service provider do not communicate with each other. Instead, they send messages to F. Each user u are initialized with a vector (of ratings) $\vec{r_u}$. The recommender and the service provider is initialized with nothing. F is parameterized with the prediction algorithm for users and for each item, and the master key S_M. Each user u sends $\vec{r_u}$ to F. These messages arrive instantly. Then F returns u the predicted ratings (E_{uj}, D_{uj}) for every item j, and returns to the recommender $Enc(S_M, r_{uj})$ for every item j, every user u to the recommender. Each user, the recommender, and the service provider output whatever F returns to them respectively.

2.4 User-level privacy

In parallel, we also consider a user-level privacy to describe the privacy (leak) of user profiles from the outputs to users, which, for example, are the recommendations received in a recommender system.

Definition 3 (User-level privacy)**.** We say that a multiparty algorithm ensures *user-level* privacy if for every algorithm that corrupts some parties in the real algorithm, there exists an algorithm that corrupts the same parties in the ideal algorithm (which needs to be defined) such that for every possible input, given inputs of all parties except one [2] non-corrupted party, it is unable to distinguish between the joint "output" of all corrupted parties in the real algorithm and that in the ideal algorithm.

The joint "output" is only composed of arbitrary information from the corrupted parties.

Remark 1. This notion of user-level privacy is more general than existing privacy definitions in that we are able to change the ideal algorithm and the indistinguishability measure to fulfill different privacy requirements.

Remark 2. System-level privacy and user-level privacy complement each other. Intuitively, system-level privacy ensures that no party could know more than what is revealed by the joint output. In other words, messages between parties should reveal no more information than the joint output. On the other hand, user-level privacy ensures that information of a particular party, revealed by the joint output to the corrupted parties, is bounded.

We measure the indistinguishability in the user-level privacy by ϵ-indistinguishability and (ϵ, δ)-indistinguishability.

Definition 4 (ϵ-indistinguishability and (ϵ, δ)-indistinguishability)**.** [4] Let X and Y be two random variables. Let $\mathcal{X}$ and $\mathcal{Y}$ be the supports of X and Y respectively.

X and Y are ϵ-indistinguishable if $\forall T \subseteq \mathcal{X} \cup \mathcal{Y}$,

$$e^{-\epsilon} Pr(Y \in T) \leq Pr(X \in T) \leq e^{\epsilon} Pr(Y \in T).$$

X and Y are (ϵ, δ)-indistinguishable if $\forall T \subseteq \mathcal{X} \cup \mathcal{Y}$,

$$Pr(X \in T) \leq e^{\epsilon} Pr(Y \in T) + \delta$$
$$\text{and } Pr(Y \in T) \leq e^{\epsilon} Pr(X \in T) + \delta.$$

Obviously, if X and Y are $(\epsilon, 0)$-indistinguishable then X and Y are ϵ-indistinguishable; and vice versa.

2.4.1 Properties of indistinguishability

We are going to show that our definition of user-level privacy is essentially equivalent to differential privacy [7] if the indistinguishability in Definition 3 is measured by the ϵ-indistinguishability defined as Definition 4. To prove the equivalence, we need the following useful properties of ϵ-indistinguishability and (ϵ, δ)-indistinguishability.

[2] Here "one" means "any one".

Lemma 1. *Let $\mathcal{A}$ be an arbitrary non-deterministic algorithm. Then if X and Y are two ϵ-indistinguishable random variables whose supports $\mathcal{X}$ and $\mathcal{Y}$ fall in the range of input of $\mathcal{A}$, then $\mathcal{A}(X)$ and $\mathcal{A}(Y)$ are also ϵ-indistinguishable.*

If X and Y are two $(\epsilon, \delta))$-indistinguishable random variables whose supports $\mathcal{X}$ and $\mathcal{Y}$ fall in the range of input of $\mathcal{A}$, then $\mathcal{A}(X)$ and $\mathcal{A}(Y)$ are also (ϵ, δ)-indistinguishable.

Proof. We first prove the lemma for ϵ-indistinguishability.

Let us separate the random coins r from $\mathcal{A}$ and define a deterministic algorithm $\mathcal{B}$ such that $\mathcal{B}(x, r) = \mathcal{A}(x)$ with $R = r$ as the value of the internal coin R. Let T be any subset of the output range of $\mathcal{A}$. Then

$$\begin{aligned} &Pr[\mathcal{A}(X) \in T] \\ =&\sum_{x\in\mathcal{X}}\sum_{r\in\mathcal{R}} Pr[\mathcal{B}(x,r) \in T] \cdot Pr(X = x)Pr(R = r) \\ =&\sum_{r\in\mathcal{R}} Pr(R = r) \sum_{x\in\mathcal{X}} \cdot Pr(X = x)Pr[\mathcal{B}(x,r) \in T] \end{aligned}$$

For any r, define $D(T, r) = \{x \in \mathcal{X} | \mathcal{B}(x, r) \in T\}$. Then for any $x \in D(T, r)$, $Pr[\mathcal{B}(x, r) \in T] = 1$ and for the others, $Pr[\mathcal{B}(x, r) \in T] = 0$. Then

$$Pr[\mathcal{A}(X) \in T] = \sum_{r\in\mathcal{R}} Pr(R = r)Pr[X \in D(T, r)].$$

Similarly, $Pr[\mathcal{A}(Y) \in T] = \sum_{r\in\mathcal{R}} Pr(R = r)Pr[Y \in D(T, r)]$. Therefore,

$$\begin{aligned} &Pr[\mathcal{A}(X) \in T] \\ =&\sum_{r\in\mathcal{R}} Pr(R = r)Pr[X \in D(T, r)] \\ \leq&\sum_{r\in\mathcal{R}} Pr(R = r)e^{\epsilon}Pr[Y \in D(T, r)] \\ =&e^{\epsilon}Pr[\mathcal{A}(Y) \in T] \end{aligned}$$

Similarly, $Pr[\mathcal{A}(Y) \in T] \leq Pr[\mathcal{A}(X) \in T]$. As a result, $\mathcal{A}(X)$ and $\mathcal{A}(Y)$ are also ϵ-indistinguishable.

Similarly, we are able to prove the lemma for (ϵ, δ)-indistinguishability, which is omitted. □

In other words, the application of a non-deterministic algorithm to two random variables does not increase the indistinguishability between them.

Lemma 2. *[Transitivity] Let X_1, X_2, Y be three random variables. If X_1 and Y are ϵ_1-indistinguishable, and X_2 and Y are ϵ_2-indistinguishable, then X_1 and X_2 are $(\epsilon_1 + \epsilon_2)$-indistinguishable.*

If X_1 and Y are (ϵ_1, δ_1)-indistinguishable, and X_2 and Y are (ϵ_2, δ_2)-indistinguishable, then X_1 and X_2 are $(\epsilon_1 + \epsilon_2, \max(\delta_1 e_2^{\epsilon} + \delta_2, \delta_2 e_1^{\epsilon} + \delta_1))$-indistinguishable.

Proof. We first prove the lemma for ϵ-indistinguishability.

Let $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{Y}$ be the supports of X_1, X_2 and Y respectively. Since, X_1 and Y are ϵ_1-indistinguishable, and X_2 and Y are ϵ_2-indistinguishable, then $\forall T \subseteq \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{Y}$,

$$e^{-\epsilon_1}Pr(X_1 \in T) \leq Pr(Y \in T) \leq e^{\epsilon_1}Pr(X_1 \in T);$$

$$e^{-\epsilon_2}Pr(X_2 \in T) \leq Pr(Y \in T) \leq e^{\epsilon_2}Pr(X_2 \in T).$$

Then $\forall T \subseteq \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{Y}$,

$$Pr(X_1 \in T) \leq e^{\epsilon_1}Pr(Y \in T) \leq e^{\epsilon_1+\epsilon_2}Pr(X_2 \in T);$$

$$Pr(X_2 \in T) \leq e^{\epsilon_1}Pr(Y \in T) \leq e^{\epsilon_1+\epsilon_2}Pr(X_1 \in T).$$

Therefore X_2 and Y are not $(\epsilon_2 + \epsilon_1)$-indistinguishable.

Similarly, we are able to prove the lemma for (ϵ, δ)-indistinguishability, which is omitted. □

2.4.2 Equivalence with differential privacy

A classical privacy definition is differential privacy [7]. It was proposed by Dwork and has been used in McSherry et al's solution to private recommender systems [19].

Definition 5 (ϵ-differential privacy and (ϵ, δ)-differential privacy)**.** [7] Let $\mathcal{A}$ be an algorithm. $\mathcal{A}$ has a dataset as its input. Let X_d be the random variable for the output of $\mathcal{A}$ when d is the input.

Then $\mathcal{A}$ is ϵ-differentially private if for any such d_1, d_2 that only one row differs, X_{d_1} and X_{d_2} are ϵ-indistinguishable.

$\mathcal{A}$ is (ϵ, δ)-differentially private if for any such d_1, d_2 that only one row differs, X_{d_1} and X_{d_2} are (ϵ, δ)-indistinguishable.

Remark 3. Definition 5 is about an algorithm that takes a dataset as its input but it could be easily extended to algorithms that take more inputs.

Let us define an algorithm $\mathcal{B}$ (which we call a multiparty algorithm afterwards) such that $\mathcal{B}$ takes inputs $i_1, \ldots, i_n$ from n parties. Then we could define an algorithm $\mathcal{A}$ that takes a dataset $\{(1, i_1), \ldots, (n, i_n)\}$ as its input and discuss whether $\mathcal{A}$ is differentially private.

In the remainder of the paper, we will simply discuss whether $\mathcal{B}$ is differentially private without mentioning this simple conversion.

Now we are ready to present the equivalence result.

Theorem 1. *[Equivalence with differential privacy] Let $\mathcal{F}$ be a multiparty algorithm. $\mathcal{F}$ is non-deterministic and involves n parties $P_1, \ldots, P_n$. Assume that at most $k, k < n$ parties can be corrupted and when they are corrupted, they are honest-but-curious.*

Define an ideal algorithm $\mathcal{I}$, a non-deterministic algorithm that involves the same n parties except one non-corrupted party P_{nc}. $\mathcal{I}$ is the same as $\mathcal{F}$ except that the input of P_{nc} is substituted by an arbitrary value in the range of inputs and there is no output to P_{nc}.

Define a new algorithm $\mathcal{B}_{c_1,\ldots,c_k}$ that involves the same n parties. $\mathcal{B}_{c_1,\ldots,c_k}$ is the same as $\mathcal{F}$ except that $\mathcal{B}$ only has outputs to $P_{c_1}, \ldots, P_{c_k}$.

If $\mathcal{B}_{c_1,\ldots,c_k}$ is ϵ-differentially private (or (ϵ, δ)-differentially private) for every $\{c_1, \ldots, c_k\} \subset [n]$, then the multiparty algorithm $\mathcal{F}$ is private with respect to the ideal algorithm $\mathcal{I}$ and ϵ-indistinguishability (or (ϵ, δ)-indistinguishability) defined by Definition 4.

If the multiparty algorithm $\mathcal{F}$ is private with respect to the ideal algorithm $\mathcal{I}$ and the $\epsilon/2$-indistinguishability (or $(\epsilon/2, \frac{1}{1+e^{\epsilon/2}}\delta)$-indistinguishability) defined by Definition 4, then $\mathcal{B}_{c_1,\ldots,c_k}$ is ϵ-differentially private (or (ϵ, δ)-differentially private) for every $\{c_1, \ldots, c_k\} \subset [n]$.

Proof. We prove the theorem for ϵ-indistinguishability. Similarly, we are able to prove the lemma for (ϵ, δ)-indistinguishability, which is omitted.

(From differential privacy to multiparty privacy.) For every algorithm $\mathcal{A}$ that corrupts k parties in $\mathcal{F}$, we use the same algorithm $\mathcal{A}$ to corrupt the same k parties in $\mathcal{I}$. Let $c_1, \ldots, c_k \subset [n]$ denote the indices of corrupted parties. Let $\underline{i_c} = i_{c_1}, \ldots, i_{c_k}$ be the inputs to the corrupted party.

Since those corrupted parties are honest-but-curious, by Definition 4, we need to prove that for every $c_1, \ldots, c_k \subset [n]$, for every $i_1, \ldots, i_n$ and nc,

$$X = \mathcal{A}(\underline{i_c}, Z_1) \text{ where } Z_1 = \mathcal{B}(i_1, \ldots, i_n),$$

and

$$Y = \mathcal{A}(\underline{i_c}, Z_2)$$
$$\text{where } Z_2 = \mathcal{B}(i_1, \ldots, i_{nc-1}, v, i_{nc+1}, \ldots, i_n),$$

are ϵ-indistinguishable given the input $i_1, \ldots, i_{nc-1}, i_{nc+1}, \ldots, i_n$.

Since $\mathcal{B}_{c_1,\ldots,c_k}$ is ϵ-differentially private, then by Definition 5, Z_1 and Z_2 are ϵ-indistinguishable for every $i_1, \ldots, i_n$ and nc. Then by Lemma 1 X and Y are indeed ϵ-indistinguishable for every $i_1, \ldots, i_n$ and nc and it is true for every $c_1, \ldots, c_k \subset [n]$.

Therefore $\mathcal{F}$ is private.

(From multiparty privacy to differential privacy.)

By contradiction. Suppose $\mathcal{B}_{c_1,\ldots,c_k}$ is not ϵ-differentially private for some $c_1, \ldots, c_k \subset [n]$. Then for some input $d_1 = i_1, \ldots, i_n$ and some row nc, there is at least one value v such that $\mathcal{B}(d_1)$ and $\mathcal{B}(d_2)$ are not ϵ-indistinguishable where $d_2 = i_1, \ldots, i_{nc-1}, v, i_{nc+1}, \ldots, i_n$.

Consider a specific algorithm $\mathcal{A}$ that corrupts $P_{c_1}, \ldots, P_{c_k}$. $\mathcal{A}$ just outputs whatever inputs. That is: $\underline{i_c}, \mathcal{B}(d)$ for some input $d = i_1, \ldots, i_{nc-1}, a, i_{nc+1}, \ldots, i_n$ for some a.

Let d_3 be some input that differs from d only on i_{nc}. By Definition 3, there exists an algorithm $\mathcal{A}^*$ that corrupts the same parties such that for every d, $X = [\underline{i_c}, \mathcal{B}(d)]$ and $Y = \mathcal{A}^*[\underline{i_c}, \mathcal{B}(d_3)]$ are $\epsilon/2$-indistinguishable given the input $i_1, \ldots, i_{nc-1}, i_{nc+1}, \ldots, i_n$.

Therefore, for d_1, $X_1 = [\underline{i_c}, \mathcal{B}(d_1)]$ and $Y = \mathcal{A}^*[\underline{i_c}, \mathcal{B}(d_3)]$ are $\epsilon/2$-indistinguishable. Then by Lemma 2, $X_2 = [\underline{i_c}, \mathcal{B}(d_2)]$ and Y cannot be $\epsilon/2$-indistinguishable. (Otherwise, X_1 and X_2 would be ϵ-indistinguishable.)

Since in the ideal algorithm, $\mathcal{A}^*$ is unaware of the change of inputs (from d_1 to d_2), $\mathcal{A}^*$ outputs the same Y while in the real algorithm, $\mathcal{A}$ outputs X_1 and X_2 respectively. Then X_2 and Y must be $\epsilon/2$-indistinguishable. We reach a contradiction.

Therefore $\mathcal{B}_{c_1,\ldots,c_k}$ is ϵ-differentially private for every $c_1, \ldots, c_k \subset [n]$. □

3 Homomorphic encryption scheme: *X-HE*

3.1 Background

Our homomorphic encryption scheme *X-HE*, inspired by Zhou et al's scheme on integer vectors (denoted by Z) [26], is an encryption scheme over single integers.

Fully homomorphic encryption schemes are bit-by-bit encryption schemes that support binary addition and multiplication. With binary addition and multiplication, we are able to build any Boolean function. They make those encryption schemes fully homomorphic and also inefficient. For example, to do 8-bit addition, we need to first build an 8-bit adder over encrypted bits. For the computation in real applications, we need to build large Boolean circuits.

Instead, both *X-HE* and Z are homomorphic encryption scheme over integers. Neither are fully homomorphic; they support only operations over integers, need no such Boolean circuit and hence are more efficient.

In two aspects, *X-HE* supports integer operations more effectively and efficiently than Z.

- *X-HE* allows the encryption of negative integers while Z does not. The negative integers are necessary and useful in real applications. For example, when we compute comparison, we need our encryption scheme to support negative integers. (Otherwise, comparison is not well-defined.)
- Both *X-HE* and Z allow key-switching (an algorithm that switches ciphertexts encrypted under one key to the ciphertexts under another) but *X-HE* involves multiplication with an $n\times[1+(n-1)l]$ matrix while Z, an $n\times nl$ matrix (if we use Z as an encryption scheme over single integers as well), where n is the length of ciphertext vectors and l is the bit-length of each element in a ciphertext vector.

In addition, we make *X-HE* a symmetric-key encryption, which is conceptually easier to use in building *X-Rec*, while Z is a public-key encryption scheme. Moreover, Z runs a deterministic encryption algorithm and is thus not semantically secure [11]. In other words, for some messages m_0 and m_1, an adversary is able to tell whether an encryption is an encryption of message m_0 or m_1. This is less secure than *X-HE*.

3.2 Encryption scheme

Definition 6 (HE)**.** The symmetric-key encryption scheme *X-HE* consists of the following three algorithms.

1. *Key generation (KeyGen):* Let $\vec{t}$ be uniformly chosen at random from $\mathbb{Z}_q^{n-1}$ where q, n are parameters of the scheme. The secret key $\vec{s} = (1|\vec{t})$.

2. *Encryption (Enc):* An algorithm which takes a key $\vec{s}$ and an integer $y, -q/2w \leq y \leq q/2w$ as input, and outputs a ciphertext $\vec{c} \in \mathbb{Z}_q^n$. w is a parameter of the scheme. The remainder of q by w approximates to 0. Then if $y \geq 0$, we let message $x = y$; otherwise, $x = y + q/w$.

   Choose an integer e such that $2|e| < w$ from the distribution χ_α. χ_α is a discrete distribution over $\mathbb{Z}_q$. Let $\vec{s} = (1, s_2, \ldots, s_n)$ and let $\vec{c} = (c_1, \ldots, c_n)$. Choose $c_2, \ldots, c_n$ uniformly random from $\mathbb{Z}_q$ and then put $c_1 = wx + e - \sum_{i=2}^{n} c_i s_i \mod q$.

3. *Decryption (Dec):* An algorithm which takes a key $\vec{s}$ and a ciphertext $\vec{c}$ as input, and outputs an integer y: $y = x$ if $x < q/2w$ and $y = x - q/w$ otherwise where

$$x = \lceil \frac{(\vec{s} \cdot \vec{c}) \mod q}{w} \rfloor.$$

   Here $\cdot$ is the inner product of two vectors and $\lceil \rfloor$ is the rounding of a rational number.

Correctness: For all keys $\vec{s}$ and messages x, we have

$$\begin{aligned}
&c_1 = wx + e - \sum_{i=2}^{n} c_i s_i + kq, k \in \mathbb{Z};\\
&Dec_{\vec{s}}(Enc_{\vec{s}}(x))\\
&= \lceil (\sum_{i=1}^{n} c_i s_i \mod q)/w \rfloor\\
&= \lceil (wx + e)/w \rfloor\\
&= x, \text{ if } 2|e| < w.
\end{aligned}$$

Let $\vec{c_1}$, $\vec{c_2}$ be two valid encryptions of integers, respectively, under the same key $\vec{s}$. Then *X-HE* supports the following four arithmetic operations:

- **Addition**: $\vec{c} = (\vec{c_1} + \vec{c_2}) \mod q$ where the addition is element wise.
- **Multiplication**: $\vec{c^*} = (\vec{c_1} \otimes \vec{c_2}) \mod q$, which denotes the dimension-n^2 vector whose entries are all the products of one entry from c_1 and one from c_2 (without any modular reduction). Denote $\vec{s^*} = (\vec{s_1} \otimes \vec{s_2}) \mod q$ similarly.

  Let $\vec{c}_{digits}$ be the vector that represents each integer of $\vec{c}_{round} = \lceil \vec{c^*}/w \rfloor$ in l digits (with base d, a parameter of *X-HE* and $l = \lfloor \log_d(q) \rfloor + 1$); i.e., $\vec{c}_{digits}$ can be considered as a block matrix $(\vec{c}_{digits,1}, \vec{c}_{digits,2}, \ldots, \vec{c}_{digits,l})$ where $\sum_{i=1}^{l} \vec{c}_{digits,i} \cdot d^{i-1} = \vec{c}_{round}$. Let $\vec{s}_{copies}$ be the vector that represents such l copies of $\vec{s^*}$ that $\vec{s^*} = (\vec{s}_{copies,1}, \vec{s}_{copies,2}, \ldots, \vec{s}_{copies,l})$ where $\vec{s}_{copies,i} = \vec{s^*} \cdot d^{i-1}$ for $i = 1, 2, \ldots, l$.

  Let M be the matrix such that $\vec{s}M = \vec{k}q + \vec{s}_{copies} + \vec{e}$ for some integer vector $\vec{k} \in \mathbb{Z}_q^{n^2}$ and some error vector $\vec{e} \in \mathbb{Z}_q^{n^2}$. Then $\vec{c} = \vec{c}_{digits} M^T \mod q$.
- **Product**: $\vec{c} = d \cdot \vec{c_1} \mod q$ where each element of $\vec{c_1}$ is scaled up by d.
- **Negation**: $\vec{c} = -\vec{c_1} \mod q$ where $-$ is element-wise.

Correctness: For addition, product, and negation, the correctness is straightforward. For multiplication, we have:

$$\begin{aligned} w\vec{c} \cdot \vec{s} &\approx \vec{c^*} \cdot \vec{s^*} = (\vec{c_1} \otimes \vec{c_2}) \cdot (\vec{s_1} \otimes \vec{s_2}) \\ &= (\vec{c_1}^T \vec{c_2}) \cdot (\vec{s_1}^T \vec{s_2}) \end{aligned}$$

where the last $\cdot$ is abused to denote the element-wise dot product between two matrices. Let $b_i = 1$ if $y_i < 0$ and 0 otherwise for $i = 1, 2$. Then

$$\begin{aligned} & w\vec{c} \cdot \vec{s} \\ \approx & (\vec{c_1}^T \vec{c_2}) \cdot (\vec{s_1}^T \vec{s_2}) \\ = & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{1i} c_{2j} s_{1i} s_{2j} = (\vec{c_1} \cdot \vec{s_1})(\vec{c_2} \cdot \vec{s_2}) \\ = & wKq + w^2[y_1 y_2 + (b_1 + b_2 - 2b_1 b_2)q/w] + we^* \end{aligned}$$

for some integer K and an error term

$$\begin{aligned} e^* &\approx y_1 e_2 + y_2 e_1 + e_1 e_2/w + (k_1 e_2 + k_2 e_1)q/w \\ &\approx y_1 e_2 + y_2 e_1 + e_1 e_2/w \end{aligned}$$

when $q/w \ll w$; $k_1, k_2 \in \mathbb{Z}$. Since $b_1 + b_2 - 2b_1 b_2 = 1$ if and only if $b_1 \neq b_2$, $\vec{c}$ is a valid encryption of $y_1 y_2$ if $2|e^*| < w$.

Remark 4. *X-HE* supports addition (Add), multiplication (Mul), scalar product (Prod) and negation (Neg) over ciphertexts. However, *X-HE* does not naturally support comparison.

Remark 5. The matrix M in the multiplication can be built in a same way as a key-switching matrix, shown in the next sub-subsection.

3.3 Key switching

Our encryption scheme *X-HE* also supports a key-switching operation that, given a ciphertext $\vec{c^*} \in \mathbb{Z}_q^n$ under the secret key $\vec{s^*} \in \mathbb{Z}_q^n$, produces a ciphertext $\vec{c} \in \mathbb{Z}_q^n$ under the secret key $\vec{s} \in \mathbb{Z}_q^n$.

Let $\vec{s^*} = (1|\vec{t^*})$. Let $\alpha^* = (1|\vec{t^*}|d\vec{t^*}|\ldots|d^{l-1}\vec{t^*})$ Let M be the matrix such that $\vec{s}M = \vec{k}q + \alpha^* + \vec{e}$ for some integer vector $\vec{k} \in \mathbb{Z}_q^n$ and some error vector $\vec{e} \in \mathbb{Z}_q^n$. (For each column $(m_{1j}, m_{2j}, \ldots, m_{nj})^T$ of M, we can choose $m_{2j}, \ldots, m_{nj} \in \mathbb{Z}_q$ uniformly at random and calculate $m_{1j} = \alpha_j^* + e_j - \sum_{i=2}^{n} m_{ij} s_i \mod q$ for $j = 1, \ldots, 1 + (n-1)l$.) M is called a key-switching matrix.

Let $\vec{c^*} = (c_1^*, \ldots, c_n^*)$. Let $\beta = (c_1^*|\beta_0|\beta_1|\ldots|\beta_{l-1})$ where $\sum_{i=0}^{l-1} \beta_i \cdot d^i = (c_2^*, c_3^*, \ldots, c_n^*)$. Then the key-switching matrix $M \in \mathbb{Z}_q^{n \times [1+(n-1)l]}$ allows us to produce $\vec{c} = \beta M^T \mod q$ without knowing $\vec{s}$.

3.4 General security claims

The security of this symmetric-key encryption scheme is based on the following assumption.

Assumption 1. [23] Let $\chi = \chi_\alpha$ be a discrete distribution over $\mathbb{Z}_q$. Let $A_{\vec{t},\chi}$ be the distribution on $\mathbb{Z}_q^n$ obtained by choosing a vector $\vec{a} \in \mathbb{Z}_q^{n-1}$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to χ, and outputting a sample $(\vec{a} \cdot \vec{t} + e, \vec{a})$, where additions are modulo q. Let U be the uniform distribution on $\mathbb{Z}_q^n$.

Let ℓ be the implicit security parameter. Then for an integer $q = q(\ell)$, an integer $n = n(\ell)$ and $\alpha = \alpha(\ell) \in [0, 1)$, we say that an algorithm distinguishes $A_{\vec{t},\chi}$ and U if for $\vec{t}$ uniformly chosen from $\mathbb{Z}_q^{n-1}$, given samples of R which is either $A_{\vec{t},\chi}$ or U, the acceptance probability of samples when $R = A_{\vec{t},\chi}$ and the acceptance probability of samples when $R = U$ differ by a non-negligible amount in ℓ.

We assume that there is no algorithm that can distinguish $A_{\vec{t},\chi}$ and U in time polynomial to ℓ. In other words, the two distributions $A_{\vec{t},\chi}$ and U are computationally indistinguishable.

This is also called the decisional Learning-with-Errors (LWE) problem, which is shown as hard as the shortest vector problems in lattices in [23, 21].

The security consists of two parts: (1) as a symmetric encryption, the ciphertext should be indistinguishable from random messages (Lemma 3); (2) as the key switching matrix is exposed, the matrix should be indistinguishable from random matrices (Lemma 4). Both proofs follow the ideas in the proof of security of Regev's cryptosystem [23] and are omitted.

Lemma 3. *If there exists a polynomial algorithm W that distinguishes between encryptions of m_0 and m_1 for arbitrary $m_0 \neq m_1$ then there exists a distinguisher Z that distinguishes between $A_{\vec{t},\chi}$ and U for a non-negligible fraction of all possible $\vec{t}$'s.*

Lemma 4. *Let $\vec{e}$ be sampled from χ^n. If there exists a polynomial algorithm W that distinguishes between the key switching matrix and a random matrix then there exists a distinguisher Z that distinguishes between $A_{\vec{t},\chi}$ and U for a non-negligible fraction of all possible $\vec{t}$'s.*

3.5 *X-Rec* parameters

X-Rec uses the following parameters for *X-HE*: $q = 2^{199}$, $d = 2$ (and therefore, $l = 199$), $w = 2^m$ for $m = 179$, $n = 359$, and finally χ is a uniform discrete distribution over $\mathbb{Z}_{2^k}$ where $k = 140$. In addition, elements of the ciphertext vector are odd numbers.

For this particular set of parameters, the ciphertext distinguishing problem is as hard as the key recovery problem.

Lemma 5 (Decision to search)**.** *Let $B_{\vec{t},\chi}^{l-1}$ be a distribution on $\mathbb{Z}_q^n$. $B_{\vec{t},\chi}^{l-1}$ is obtained as $A_{\vec{t},\chi}$ except that at the last step, $B_{\vec{t},\chi}^{l-1}$ outputs $(v \cdot 2^{l-1} + (\vec{a} \cdot \vec{t} + e \mod 2^{l-1}), \vec{a})$, where additions are modulo q and v is a uniform random bit.*

Assume that we have access to procedure $\mathcal{W}$ that for all $\vec{t}$ accepts with probability exponentially close to 1 on inputs from $A_{\vec{t},\chi}$ and rejects with probability exponentially close to 1 on inputs from $B_{\vec{t},\chi}^{l-1}$.

Then for $(k-1)(n-1) \leq poly(\ell)$, there exists an efficient algorithm $\mathcal{W}'$ that, given samples from $A_{\vec{t},\chi}$ for some $\vec{t}$, outputs $\vec{t}$ with probability exponentially close to 1.

Proof. We first show how $\mathcal{W}'$ finds the least but two significant bit of $t_1 \in \mathbb{Z}_q$, the first coordinate of $\vec{t}$. Finding the least significant k bits of all coordinates is similar. Let $g = 1$, consider the following transformation. Given a pair $(v \cdot 2^{l-1} + x, \vec{a})$ we output the pair $(v \cdot 2^{l-1} + x + r2^{l-2} \cdot (2g+1), \vec{a} + (r2^{l-2}, 0, \ldots, 0))$ where $r \in \mathbb{Z}_q$ is chosen uniformly at random.

Then it always takes $A_{\vec{t},\chi}$ to $B_{\vec{t},\chi}^{l-1}$ if $(t_1)_1 \neq 1$; and it takes $A_{\vec{t},\chi}$ to itself if $(t_1)_1 = 1$. (Note that since the elements of the ciphertext vector are odd numbers, $(t_1)_0$ is always 1.) Hence, using $\mathcal{W}$, we can test whether $g = (t_1)_1$. We need only to call $\mathcal{W}$ for $(k-1)(n-1) < poly(\ell)$ times to find the least significant k bits of all coordinates.

After we have found the least significant k bits of all coordinates of $\vec{t}$, the error vectors used in creating the samples of $A_{\vec{t},\chi}$ are determined. Thus with $n-1$ samples on average, we are able to solve $\vec{t}$ using Gaussian elimination. □

Thus, if the key recovery problem is hard, then there is no efficient algorithm that distinguishes between $B_{\vec{t},\chi}^{l-1}$ and $A_{\vec{t},\chi}$.

Let us define more distributions. We denote by $B_{\vec{t},\chi}^{i}$ the distribution obtained by drawing samples from $A_{\vec{t},\chi}$ and outputting $v \cdot 2^i + (\vec{a} \cdot \vec{t} + e \mod 2^i, \vec{a})$ where $v \in \mathbb{Z}_{2^i}$ is uniformly chosen at random.

We compare $B_{\vec{t},\chi}^{l-1}$ and $B_{\vec{t},\chi}^{l-2}$. Similar to Lemma 5, if the key recovery problem is hard, then there is no efficient algorithm that distinguishes between $B_{\vec{t},\chi}^{l-1}$ and $B_{\vec{t},\chi}^{l-2}$. In fact, for $i = l-1, \ldots, m+1$, if the key recovery problem is hard, there is no efficient algorithm that distinguishes between $B_{\vec{t},\chi}^{i}$

and $B^{i-1}_{\vec{t},\chi}$. Therefore, there is no efficient algorithm that distinguishes between $B^{m}_{\vec{t},\chi}$ and $A_{\vec{t},\chi}$.

For encryption, *X-HE* indeed embeds the integer y in the most significant $l-m$ bits of $(\vec{a}\cdot\vec{t}+e)$. Now that these bits are indistinguishable from a random string of the same length, then the encryption of any integer is indistinguishable from the encryption of random messages.

For the concrete hardness of the key recovery problem, we list the complexity of three state-of-art algorithms [3]. We assume that the adversary which tries to recover the key has access to plaintext-ciphertext pairs. In other words, the adversary is given samples from $A_{\vec{t},\chi}$.

Our final goal is to show the concrete hardness of the ciphertext distinguishing problem; therefore, we also present the time estiamte and the distinguishing advantage of the state-of-art algorithm [20, 24, 17] that performs the distinguisher attack directly. We assume that the adversary which tries to distinguish the ciphertext from a random string has access to samples of one kind. In other words, the adversary is given samples from either $A_{\vec{t},\chi}$ or u.

- **Exhaustive search** [3]: The exhaustive search guesses the errors for the first $n-1$ samples and for each error, checks whether they give a correct error for the next $n-1$ samples. When the errors are determined, Gaussian elimination is applied to solve $\vec{t}$. The time complexity with the exhaustive search is $(2^k)^{n-1}\cdot 2(n-1)$. The memory complexity is $n-1$. The sample complexity is $2(n-1)$.

- **Meet-in-the-middle** [3]: This is also an exhaustive search. However, when guessing the errors for the first $n-1$ samples, the meet-in-the-middle attack splits the error in half and searches samples for collisions, i.e. a combination of the first half and the second half that gives a correct error in the range of the next $(n-1)$ samples. The time complexity with the meet-in-the-middle attack is $(2^k)^{\lceil (n-1)/2\rceil}\cdot 2(n-1)$. The memory complexity is $(2^k)^{\lceil (n-1)/2\rceil}$. The sample complexity is $2(n-1)$.

- **BKW** [2]: This attack tries to find a short vector in $\mathbb{Z}_q^{n-1}$ and then solves $\vec{t}$. The attack splits the vector of $\vec{a}$ into a blocks, each of width b, and searches for collisions in, for example, the first block, to cancel that block. For the appropriate choice of a and b such that $n-1\leq ab$, the time complexity with BKW is at least $(\frac{q^b-1}{2})\cdot(\frac{a(a-1)}{2}n-\frac{ba(a-1)}{4}-\frac{b}{6}((a-1)^3+\frac{3}{2}(a-1)^2+\frac{1}{2}(a-1)))$. The memory complexity is $\frac{q^b}{2}\cdot a\cdot(n-b\frac{a-1}{2})$. The sample complexity is $aq^b/2$.

- **Basic distinguisher attack** [20, 24, 17]: This attack writes m samples as an $m\times n$ matrix $(\vec{x}^T, A)$, calculates $val=\frac{1}{q}(\vec{b}\cdot\vec{w} \mod q)$ with a short $\vec{w}$ such that $A\vec{w}^T=\vec{0} \mod q$ and decides on $A_{\vec{t},\chi}$ if $v\in[-0.25, 0.25]$ (and U otherwise). So far, no published work gives a concrete analysis of the distinguishing advantage of this attack, which we show as follows.

$$\epsilon(\vec{w}) = 0.5\cdot|-0.5+ \int_{-0.25}^{0.25}\sum_{j=-\infty}^{\infty}\frac{1}{\beta}\exp(-\pi(\frac{r-j}{\beta})^2)dr| \quad (1)$$

  where $\beta=\frac{\sqrt{2\pi}}{\sqrt{3}}||\vec{w}||2^{k-l}$. Combined with Lindner and Peikert's conservative estimate of the $\log_2$ runtime t in seconds of finding a short $\vec{w}$ on a single 2.3GHz AMD Opeteron machine [17], we have a relation among ϵ, t, k, l and n:

$$\epsilon = 692.39\cdot\exp(-9.8356\beta\cdot 6.4232\sqrt{(n-1)l/(t+110)})$$

  after approximating Equation 1 for $\epsilon < 0.00025$.

With a little calculation, we know that none of the first three algorithms could break the key recovery problem (or the LWE problem) in less than 2^{99} operations under our particular choice of parameters. Thus the key recovery problem is hard and the ciphertext is indistinguishable from random messages for any efficient adversary that is limited to do $2^{99}/[(k-1)(n-1)(l-m)]\approx 2^{80}$ operations.

Again with some calculation, we know that the last algorithm could not break the ciphertext distinguishing problem (or the decision LWE problem) in less than $2^{33.038}$ seconds for a distinguishing advantage 2^{-32} under our choice of parameters. As mentioned in Lenstra' s discussion about the key sizes [16], if we assume that the machine consumes

390$, then this distinguisher attack costs at least 39.9M dollardays, equivalent to the cost of breaking a block cipher with a 80-bit key which approximately requires 2^{80} operations.

Then we conclude that for *X-HE*, the ciphertext is indistinguishable from random messages for any efficient adversary that is limited to do 2^{80} operations.

4 *X-Rec* recommendation algorithms

4.1 Online recommendation

Recall that there are three main parties involved in *X-Rec*: a trusted user proxy, an untrusted recommender, and an untrusted service provider. There are two keys used in online recommendation in *X-Rec*: S_M and S_G. Both are generated by the trusted party at the one-time setup of *X-Rec*. The untrusted recommender stores S_G and, whenever a user is online, sends S_G to the user. S_M is never distributed to any party but the trusted party provides two matrices Mt_{S_M,S_G} and Mt_{S_G,S_M} to help switch two keys and distributes them to the untrusted service provider and the trusted proxy respectively.[3]

In *X-Rec*, to compute recommendations means to compute a predicted rating for every movie item, order them locally and then present them to users. The profiles are locally stored. One profile is a vector of integer values $\vec{r_u} = (r_{u1}, r_{u2}, \ldots, r_{uL})$ for L movie items. One prediction/recommendation is a vector of pairs of integer values: $\vec{P_u} = ((E_{u1}, D_{u1}), (E_{u2}, D_{u2}), \ldots, (E_{uL}, D_{uL}))$. Then the online recommendation goes as follows.

1. The trusted proxy encrypts the user profile encrypted under key S_G:

$$Enc(S_G, r_{u1}), \ldots, Enc(S_G, r_{uL}).$$

2. The trusted proxy switches the encrypted profile from key S_G to key S_M (using Mt_{S_G,S_M}) and sends the user profile encrypted under key S_M to the untrusted recommender:

$$Enc(S_M, r_{u1}), \ldots, Enc(S_M, r_{uL}).$$

3. The untrusted recommender computes the predicted ratings encrypted under key S_M. First the recommender samples uniformly from all user profiles, and then runs X-NN (explained later) with the service provider, and finally predicts ratings based on sampled users for the user u according to Equation (2).

4. The untrusted recommender sends the predicted ratings P (masked with randomness R) encrypted under key S_M to the untrusted service provider:

$$(Enc(S_M, E_{u1} + R), Enc(S_M, D_{u1} + R)),$$

$$\ldots, (Enc(S_M, E_{uL} + R), Enc(S_M, D_{uL} + R));$$

and sends R to the trusted proxy.

5. The untrusted service provider switches the encrypted predicted ratings from key S_M to key S_G (using Mt_{S_M,S_G}) and sends the predicted ratings encrypted under S_G to the trusted proxy:

$$(Enc(S_G, E_{u1} + R), Enc(S_G, D_{u1} + R)),$$

$$\ldots, (Enc(S_G, E_{uL} + R), Enc(S_G, D_{uL} + R)).$$

4.2 Comparison protocol *X-NN*

Recall that the encryption scheme implemented in *X-Rec* is an homomorphic encryption scheme over integers, which does not support comparison as an operator. (However, *X-HE* supports addition, multiplication, subtraction and scalar product, which will be used in building an comparison.) Thus in the online recommendation, the recommender needs to communicate with the service provider to finish comparison. To do comparison $x > 0$ over ciphertexts means given c, a ciphertext of x with key S_M, to find ζ such that ζ is a ciphertext of $(x > 0)$ with key S_M; for other comparison $x > C$, we first do homomorphic subtraction $x - C$ and then do the comparison $x - C > 0$. Note that although x is the integer considered, the plaintext of c is p which represents x in a certain way.

[3] For simplicity, item sampling is not considered in the recommendation algorithm. Moreover, in *X-Rec*, the prediction/recommendation is actually calculated offline and periodically, of which the system-level privacy is equivalent to this online version. The online version is considered here for the ease of presentation.

There are three keys involved in the comparison: S_M, S_G and S_H. S_H is also generated and distributed to the untrusted service provider by the trusted party at the one-time setup. In addition, Mt_{S_G,S_M}, Mt_{S_H,S_M} and Mt_{S_M,S_H} are generated and distributed to the untrusted recommender by the trusted party as well.

Recall that the plaintext space of *X-HE* is $\mathbb{Z}_{q/w}$. Assume that $q/w = 2^l$ for some integer l. Then the comparison goes as follows. (This can be easily extended to the more general integer q/w and is thus omitted.)

1. The recommender generates l random bits and encrypts them under key S_G:

$$Enc(S_G, a_0), \ldots, Enc(S_G, a_{l-1});$$

$$Enc(S_G, a) \text{ where } a = a_{l-1} \ldots a_0 \in \{0,1\}^l.$$

2. The recommender switches the encrypted random bits from key S_G to key S_M:

$$\gamma_0 = Enc(S_M, a_0), \ldots, \gamma_{l-1} = Enc(S_M, a_{l-1});$$

$$\gamma = Enc(S_M, a)$$

and masks x with a:

$$Add(c + \gamma).$$

3. The recommender again switches the encrypted $p + a$ from key S_M to S_H and sends the masked integer encrypted under S_H to the service provider:

$$\omega = Enc(S_H, p + a \mod q/w).$$

4. The service provider decrypts ω, decomposes the integer into bits and re-encrypts every bit under key S_H and sends them to the recommender:

$$Enc(S_H, y_0), \ldots, Enc(S_H, y_{l-1})$$

$$\text{where } p+a \mod q/w = y = y_{l-1} \ldots y_0 \in \{0,1\}^l.$$

5. The recommender switches those encrypted bit from key S_H to key S_M:

$$\alpha_0 = Enc(S_M, y_0), \ldots, \alpha_{l-1} = Enc(S_M, y_{l-1}).$$

6. Then the recommender calculates $\kappa_0, \ldots, \kappa_{l-1}$ and ζ as follows:

$$\kappa_i = \begin{cases} \gamma_0 + Neg(Prod(\alpha_0, r_0)), \\ \text{ if } i = 1; \\ \kappa_{i-1} + \gamma_{i-1} + Neg(Mul(\alpha_{i-1}, \kappa_{i-1})) \\ + Neg(Prod(\kappa_{i-1}, r_{i-1})) \\ + Neg(Prod(\alpha_{i-1}, r_{i-1})) \\ + Prod(Prod(Mul(\alpha_{i-1}, \kappa_{i-1}), r_{i-1}), 2), \\ \text{ if } i = 2, \ldots, l-1; \end{cases}$$

$$\begin{aligned} \zeta = {} & \alpha_{l-1} + \gamma_{l-1} + \kappa_{l-1} \\ & + Neg(Prod(Prod(\alpha_{l-1}, r_{l-1}), 2)) \\ & + Neg(Prod(Mul(\alpha_{l-1}, \kappa_{l-1}), 2)) \\ & + Neg(Prod(Prod(\kappa_{l-1}, r_{l-1}), 2)) \\ & + Prod(Mul(Prod(\kappa_{l-1}, r_{l-1}), \alpha_{l-1}), 4). \end{aligned}$$

4.2.1 Correctness

Recall that with *X-HE*, a negative integer x is represented by a plaintext $p = x + q/w$ and a positive integer x, a plaintext $p = x$, and then encrypted. In other words, comparing x and 0 is equivalent to comparing p and $q/2w$. Given that $q/w = 2^l$, thus

$$z = (p)_{l-1} = \begin{cases} 0 & \text{if } x \geq 0 \\ 1 & \text{if } x < 0 \end{cases}$$

where $(p)_{l-1}$ is the $(l-1)$th bit of p. Since $y = p+a \mod q/w$ where $y, p, a \in \mathbb{Z}_{q/w}$, $p = y + \beta q/w - a$ where $\beta = 1$ if $y \geq a$; and 0 otherwise. As a result, the recommender has the encryption of bits of a and that of y. The recommender is able to discover the encryption of bit $(p)_{l-1}$. Since *X-HE* supports addition, multiplication, negation and scalar product, we only need to show the correctness of method in the plaintext space.

Let us first consider the case where $y \geq a$. $p = y - a$. Let c_i be the borrow from the ith bit of y to $(i-1)$th bit. I.e. following the textbook subtraction algorithm, $c_1 = \bar{y_0} a_0$ and $c_i = \bar{y_{i-1}} \bar{c_{i-1}} a_{i-1} \vee \bar{y_{i-1}} c_{i-1} \bar{a_{i-1}} \vee y_{i-1} c_{i-1} a_{i-1}$, $i = 2, \ldots, l-1$ where all operations are logical operations over bits. It is easy to convert

the formula above to:

$$c_i = \begin{cases} a_0 - a_0 y_0 & \text{if } i = 1 \\ c_{i-1} + a_{i-1} - y_{i-1} c_{i-1} & \\ -c_{i-1} a_{i-1} - a_{i-1} y_{i-1} & \\ +2 y_{i-1} c_{i-1} r_{i-1} & \text{if } i = 2, \ldots, l-1. \end{cases}$$

where all operations are arithmetic operations over integers supported by *X-HE*.

Similarly, since $p_{l-1} = y_{l-1} \oplus a_{l-1} \oplus c_{l-1}$, this could also be transformed into arithmetic operations as follows.

$$x_{l-1} = y_{l-1} + a_{l-1} + c_{l-1} - 2y_{l-1}a_{l-1} \\ - 2a_{l-1}c_{l-1} - 2y_{l-1}c_{l-1} + 4y_{l-1}a_{l-1}c_{l-1}.$$

Next consider $y < a$. Then $p = y + q/w - a$. $(p)_{l-1}$ can be computed similarly except that now we need to take $(q/w)_i, i = 0, \ldots, l-1$ into account for each of the previous equations. Since $(q/w)_i = 0, i = 0, \ldots, l-1$, the previous equations do not change.

Therefore, no matter whether $y < a$ or not, the recommender follows the same algorithm to calculate encrypted $(p)_{l-1}$ with $z = (p)_{l-1}$ encrypted in ζ, c_i encrypted in κ_i, y_i encrypted in α_i, and a_i encrypted in γ_i.

4.3 Prediction

Recall that in *X-Rec*, to compute recommendations means to compute a predicted rating for every movie item, order them locally and then present them to users. The formula to predict rating for some movie item j is

$$P_{u,j} = \bar{r_u} + \frac{\sum_{v \in U_j} \tau(u,v)(r_{v,j} - \bar{r_v})}{\sum_{v \in U_j} |\tau(u,v)|}$$

where U_j is the set of users who have rated item j, $\tau(u,v)$ is some similarity measure between $\vec{r_u}$ and $\vec{r_v}$, and $\bar{r_u}$ is the average rating given by user u. An example of $\tau(u,v)$ is the cosine similarity with a threshold: $\tau(u,v)$ is the similarity value if it is higher than threshold or zero otherwise. *X-Rec* ignores $\bar{r_u}$ and only computes the fraction with numerator E and denominator D, since only the order is needed.

Moreover, to reduce latency, *X-Rec* also samples users. Finally, the prediction goes as follows.

Algorithm 1

Input: a vector $\vec{r_u}$ of ratings from each of the N users;

Output: predicted ratings $(E_{uj}, D_{uj}), j = 1, \ldots, L$ of L movies to each of the N users;

1. The recommender chooses $F \cdot N$ users uniformly at random where F is a parameter for user sampling in *X-Rec*.

2. The recommender computes, for $j = 1, \ldots, L$,

$$\begin{cases} E_{uj} = \sum_{v \in U_j \cap S} \tau(u,v)(r_{v,j} - \bar{r_v}) \\ D_{uj} = \sum_{v \in U_j \cap S} |\tau(u,v)| \end{cases} \quad (2)$$

   where S is the set of FN users sampled in the previous step.

Recall that *X-HE* supports addition, multiplication, negation and scalar product. With *X-NN* in the previous subsection, the recommender is able to compute Equation 2 over encrypted ratings.

In Section 5, we will show that this prediction algorithm brings us some privacy.

5 Privacy guarantees

Recall that both system-level and user-level privacy definitions essentially say that one system is as private as the ideal system defined by us.

We will follow this path in this section. First, we define the ideal system. Second, we show that *X-Rec* ensures system-level and user-level privacy with respect to the ideal system. Third, we check whether the ideal system meets the overall privacy requirements. Since in the ideal system, the secrecy of user profiles is protected, then in *X-Rec*, it is also protected.

To prove system-level privacy and user-level privacy, we will first assume that an adversarial algorithm corrupts at most one party. We will then proceed to the case where the adversary corrupts two or more parties, i.e. collusion between parties.

5.1 System-level privacy

Formally, the privacy guarantee of *X-Rec* on the system level is:

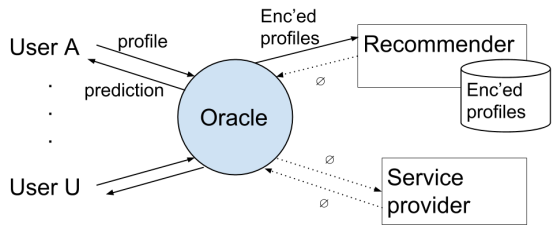

Figure 2: The ideal recommendation

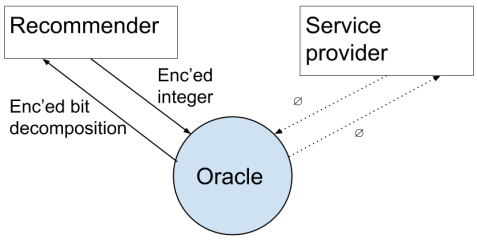

Figure 3: The ideal comparison

Theorem 2. *The online recommendation presented in Section 4 ensures system-level privacy with respect to the ideal recommendation defined in Definition 2.*

Before proving the system-level privacy, we need to first prove that *X-NN* is also a private system. I.e., the *X-NN* presented in Section 4 ensures system-level privacy with respect to the ideal comparison defined in Definition 7.

Definition 7 (Ideal comparison)**.** The ideal comparison system is composed of the recommender, the service provider, and an additional party: U. The recommender and the service provider do not communicate with each other. Instead, they send messages to U. The recommender is initialized with an encrypted integer $Enc(S_H, y)$ of integer y. U is parameterized with key S_H. The recommender sends the encrypted integer to U. U returns the recommender the encrypted bit decomposition $Enc(S_H, y_j), j = 0, \ldots, l-1$ of integer y. Messages are delivered instantly. The recommender, and the service provider outputs whatever F returns.

We recall ideal recommendation in Figure 2 and depict ideal comparison in Figure 3.

We prove the guarantee for online recommendation and *X-NN* separately. In the proof for online recommendation, we assume that *X-NN* achieves system-level privacy, and thus the *X-NN* protocol is replaced with the ideal comparison system.

System-level privacy of online recommendation.

Proof. For every algorithm $\mathcal{A}$ that corrupts the proxy of a user u, we construct an algorithm $\mathcal{A}^*$ that corrupts the same user. More specifically, $\mathcal{A}^*$ runs $\mathcal{A}$ as a black-box and plays the other parties in the online recommendation system with $\mathcal{A}$.

1. $\mathcal{A}^*$ generates keys $\hat{S}_G$, $\hat{S}_M$ and key-switching matrix $Mt_{\hat{S}_G,\hat{S}_M}$ and initializes $\mathcal{A}$ with u's profile, $\hat{S}_G$ and $Mt_{\hat{S}_G,\hat{S}_M}$.
2. $\mathcal{A}^*$ sends u's profile to F;
3. $\mathcal{A}^*$ receives the prediction from F:
$$(E_{u1}, D_{u1}), \ldots, (E_{uL}, D_{uL});$$
4. $\mathcal{A}^*$, on behalf of the recommender, receives $\mathcal{A}$'s encryption of u's profile (but ignores the encryption);
5. $\mathcal{A}^*$ chooses an integer R uniformly at random, encrypts the prediction (received from F) masked with R and sends it to $\mathcal{A}$:
$$(Enc(S_G, E_{u1} + R), Enc(S_G, D_{u1} + R)),$$
$$\ldots, (Enc(S_G, E_{uL} + R), Enc(S_G, D_{uL} + R)$$
on behalf of the service provider; $\mathcal{A}^*$ also sends R to $\mathcal{A}$ on behalf of the recommender.
6. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

We note that since $\mathcal{A}$ is honest-but-curious, $\mathcal{A}$ does not change u's profile, and therefore $\mathcal{A}^*$ does not need to wait for $\mathcal{A}$'s encrypted message. The transcript between $\mathcal{A}$ and $\mathcal{A}^*$ is computationally indistinguishable from the real transcript by Lemma 3 and Lemma 4. As a result, the joint output of the ideal recommendation system and that of the online recommendation system of *X-Rec* are computationally indistinguishable.

For every algorithm $\mathcal{A}$ that corrupts the recommender, we construct an algorithm $\mathcal{A}^*$ that also

corrupts the recommender. More specifically, $\mathcal{A}^*$ runs $\mathcal{A}$ as a black-box and plays the other parties in the online recommendation system with $\mathcal{A}$.

1. $\mathcal{A}^*$ initializes the recommender with a random matrix as the multiplication matrix Mt_mult (for the homomorphic multiplication) and a random matrix $\hat{Mt}_{S_G,S_M}$;
2. $\mathcal{A}^*$ receives profiles encrypted under S_M from F;
3. $\mathcal{A}^*$ sends encrypted profiles to $\mathcal{A}$ on behalf of all the users;
4. $\mathcal{A}^*$, on behalf of the service provider, runs the ideal comparison system with $\mathcal{A}$;
5. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

It is easy to see that the transcript between $\mathcal{A}^*$ and $\mathcal{A}$ and the real transcript between $\mathcal{A}$ and all other parties are computationally indistinguishable by Lemma 4. Since $\mathcal{A}$ is honest-but-curious, the joint output of $\mathcal{A}$ and all other parties, and the joint output of $\mathcal{A}^*$ and all other parties are computationally indistinguishable.

Finally, for every algorithm $\mathcal{A}$ that corrupts the service provider, we construct an algorithm $\mathcal{A}^*$ that also corrupts the service provider as follows.

1. $\mathcal{A}^*$ initializes $\mathcal{A}$ with a random matrix $\hat{Mt}_{S_M,S_G}$;
2. $\mathcal{A}^*$, on behalf of the recommender, sends random ciphertexts to $\mathcal{A}$;
3. $\mathcal{A}^*$, on behalf of the recommender, runs the ideal comparison with $\mathcal{A}$;
4. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

Note that in the real recommendation, the service provider only receives ciphertexts of random values (since predicted ratings are masked with uniformly random integers). By Definition 6, the distribution of random ciphertexts is computationally indistinguishable from that of ciphertexts of random values. In addition, by Lemma 4, the random matrix is computationally indistinguishable from the real key-switching matrix. Since $\mathcal{A}$ is honest-but-curious, the joint output of $\mathcal{A}$ and all other parties, and the joint output of $\mathcal{A}^*$ and all other parties are computationally indistinguishable.

Therefore the online recommendation presented in Section 4 ensures system-level privacy with respect to the ideal recommendation defined in Definition 2. □

System-level privacy of *X-NN*. Here we show the system-level privacy of X-NN (and thus in the proof of *X-Rec*, we can replace X-NN with an ideal comparison system).

Proof. For every algorithm $\mathcal{A}$ that corrupts the recommender, we construct an algorithm $\mathcal{A}^*$ that also corrupts the recommender as follows.

1. $\mathcal{A}^*$ generates three keys: $\hat{S}_G, \hat{S}_M, \hat{S}_H$;
2. $\mathcal{A}^*$ initializes $\mathcal{A}$ with a random ciphertext encrypted with $\hat{S}_M$ and matrices $Mt_{\hat{S}_G,\hat{S}_M}, Mt_{\hat{S}_H,\hat{S}_M}, Mt_{\hat{S}_M,\hat{S}_H}$;
3. $\mathcal{A}^*$ receives an encryption $w = Enc(\hat{S}_H, y)$ of some integer y from $\mathcal{A}$;
4. $\mathcal{A}^*$ decrypts w and sends the encrypted bit decomposition of y to $\mathcal{A}$:
$$Enc(\hat{S}_H, y_0), \ldots, Enc(\hat{S}_H, y_{l-1});$$
5. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

It is easy to see that the transcript between $\mathcal{A}^*$ and $\mathcal{A}$ is computationally indistinguishable from the transcript between the real service provider and $\mathcal{A}$. As $\mathcal{A}$ is honest-but-curious, then the joint output between $\mathcal{A}$ and the service provider, and the joint output between $\mathcal{A}^*$ and the service provider are computationally indistinguishable.

For every algorithm $\mathcal{A}$ that corrupts the service provider, we construct an algorithm $\mathcal{A}^*$ that also corrupts the service provider as follows.

1. $\mathcal{A}^*$ generates key $\hat{S}_H$ and initializes $\mathcal{A}$ with $\hat{S}_H$;
2. $\mathcal{A}^*$ sends a random ciphertext to $\mathcal{A}$;
3. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

Note that in *X-NN*, the service provider only receives ciphertexts of random values (since the integer y is an addition of the number to be compared and an uniformly random integer). By Definition 6, the distribution of random ciphertexts is computationally indistinguishable from that of ciphertexts

of random values. As a result, the joint output is indistinguishable between *X-NN* and the ideal comparison.

Therefore *X-NN* presented in Section 4 ensures system-level privacy with respect to the ideal comparison defined in Definition 7. □

5.2 User-level privacy

Definition 8 (Ideal prediction)**.** Let $N+1$ be the current number of users. The ideal prediction takes a vector of ratings from all $N+1$ users except an arbitrary one. The ideal prediction works the same as Algorithm 1 except that it is only applied to the N users.

Formally, the privacy guarantee of *X-Rec* on the user level is:

Theorem 3. *The online prediction presented in Section 4 ensures user-level privacy with respect to the ideal prediction defined in Definition 8 where the indistinguishability of its privacy is measured by the* $(\ln \frac{1}{1-F}, F)$*-indistinguishability.*

Recall that we assume only one corrupted user u which is honest-but-curious. Define a new algorithm $\mathcal{B}_u$ that involves the same n parties. $\mathcal{B}_u$ works the same as Algorithm 1 except that $\mathcal{B}_u$ only has an output to u. Denote by d_{N+1} an input to $\mathcal{B}_u$ that contains $N+1$ user profiles. Denote by d an input such that d_{N+1} contains exactly one more user profile than d. By Theorem 1, if the following lemma holds, then our privacy claim is true.

Lemma 6. *For all* d_{N+1}, d*,* $\mathcal{B}_u(d_{N+1})$ *and* $\mathcal{B}_u(d)$ *are* $(\ln \frac{1}{1-F}, F)$ *– indistinguishable for all* u*.*

Proof. Let $\vec{v}$ be the user profile that is not included in d. Let $s = F(N+1) \approx FN$. Denote $X_{N+1} = \mathcal{B}_u(d_{N+1})$ and $X = \mathcal{B}_u(d)$. Let $\mathcal{X}$ be the support of X. Let $\mathcal{X}_{N+1}$ be the support of X_{N+1}.

Now denote by $X(S)$ the realization of X when Algorithm 1 chooses S as the set of s sampled users. Then the multiset $\Pi = \{X(S)|S \subset d, |S| = s\}$ contains all realizations. Similarly define the multiset $\Pi_{N+1} = \{X_{N+1}(S)|S \subset d_{n+1}, |S| = s\}$. Let Δ be the multiset $\Delta = \{\mathcal{X}(S)|v \in S, S \subset [N+1], |S| = s\}$. Then Π_{N+1} is the disjoint union of Π and Δ.

Clearly, $\mathcal{X}$ and $\mathcal{X}_{N+1}$ are the corresponding set of Π and Π_{N+1} respectively. Let $\mathcal{Y}$ be the corresponding set of Δ. $\mathcal{X}_{N+1} = \mathcal{X} \cup \mathcal{Y}$.

Then we are able to divide $\mathcal{X}_{N+1}$ into three mutually disjoint subsets: $T_1 = \mathcal{Y} \backslash \mathcal{X}$, $T_2 = \mathcal{Y} \cap \mathcal{X}$ and $T_3 = \mathcal{X} \backslash \mathcal{Y}$. Clearly, $T_2 \cup T_3 = \mathcal{X}$. For any nonempty event T, at least one of the three intersections $T \cap T_1$, $T \cap T_2$ and $T \cap T_3$ is nonempty.

Let $\rho(x), q(x)$ be the proportion of x in the multiset Δ, Π respectively. Denote the cardinality of Π, Δ by b, a respectively. Then the probability distribution of X_{N+1} is:

$$Pr(X_{N+1} = x) = \frac{1}{\binom{N+1}{s}} a\rho(x),$$

when $x \in T_1$;

$$\begin{aligned} &Pr(X_{N+1} = x) \\ =&\frac{1}{\binom{N+1}{s}}[bq(x) + a\rho(x)] \end{aligned}$$

when $x \in T_2$; and

$$Pr(X_{N+1} = x) = \frac{1}{\binom{N+1}{s}} bq(x),$$

when $x \in T_3$.

Similarly, the probability distribution of X is:

$$Pr(X = x) = \frac{1}{\binom{N}{s}} bq(x)$$

when $x \in T_2 \cup T_3$; $Pr(X = x) = 0$ otherwise.

For any event T, let $T_\rho = T \cap (T_1 \cup T_2)$ and $T_q = T \cap (T_2 \cup T_3)$; then

$$\begin{aligned} &Pr(X_{N+1} \in T) \\ =&\sum_{x \in T_q} \frac{1}{\binom{N+1}{s}} bq(x) + \sum_{x \in T_\rho} \frac{1}{\binom{N+1}{s}} a\rho(x) \\ =&\frac{N+1-s}{N+1} \sum_{x \in T_q} q(x) + \frac{s}{N+1} \sum_{x \in T_\rho} \rho(x) \\ =&\frac{N+1-s}{N+1} Pr(X \in T) + \frac{s}{N+1} \sum_{x \in T_\rho} \rho(x) \end{aligned}$$

The second equality results from the fact that $b = \binom{N}{s}$ and $a = \binom{N}{s-1}$; the third equality, the fact that $T \cap \mathcal{X} \subseteq T_q$.

Since (1) $0 \leq \sum_{x \in T_\rho} \rho(x) \leq 1$ and (2) $\frac{N+1}{N+1-s} > \frac{N+1-s}{N+1}$, therefore,

$$\begin{aligned} &Pr(X_{N+1} \in T) \\ \leq&\frac{N+1-s}{N+1} Pr(X \in T) + \frac{s}{N+1} \qquad (3) \\ \leq&\frac{N+1}{N+1-s} Pr(X \in T) + \frac{s}{N+1}; \end{aligned}$$

$$
\begin{aligned}
& Pr(X \in T) \\
=& \frac{N+1}{N+1-s} Pr(X_{N+1} \in T) - c \sum_{x \in T_\rho} \rho(x) \qquad (4) \\
\leq& \frac{N+1}{N+1-s} Pr(X_{N+1} \in T)
\end{aligned}
$$

where $c = \frac{s}{N+1-s}$. Since Equation (3) and Equation (4) are true for all d_{N+1}, d and u, therefore $X = \mathcal{B}_u(d)$ and $X_{N+1} = \mathcal{B}_u(d_{N+1})$ are (ϵ, δ)-indistinguishable where $\epsilon = \ln \frac{N+1}{N+1-s} \approx \ln \frac{1}{1-F}$ and $\delta = \frac{s}{N+1} \approx F$. □

5.3 Collusion

In this subsection, we allow collusion among parties. I.e. an adversarial algorithm $\mathcal{A}$ corrupts two or more parties in *X-Rec*. We will show that *X-Rec* still ensures system-level privacy and user-level privacy.

Recall that (1) corrupted parties are still honest-but-curious; and (2) $\mathcal{A}$ does not corrupt the recommender and the service provider at the same time. We provide proof sketches for system-level and user-level privacy separately as below.

5.3.1 System-level privacy

Since $\mathcal{A}$ does not corrupt the recommender and the service provider at the same time, we only need to prove the system-level privacy for online recommendation.

First, we consider the case where more than one users are corrupted and the recommender is also corrupted. For every algorithm $\mathcal{A}$ that corrupts a set of users C, and the recommender, we build an algorithm $\mathcal{A}^*$ that corrupts C and the recommender in the ideal recommendation system as follows.

1. $\mathcal{A}^*$ generates three keys: $\hat{S}_G$, $\hat{S}_M$ and $\hat{S}_H$, and three key-switching matrices: $Mt_{\hat{S}_G,\hat{S}_M}$, $\hat{Mt}_{mult}$, $Mt_{\hat{S}_G,\hat{S}_H}$ and $Mt_{\hat{S}_H,\hat{S}_M}$;
2. $\mathcal{A}^*$ initializes $\mathcal{A}$ with $Mt_{\hat{S}_G,\hat{S}_M}$ and $hatS_G$ for every corrupted user $u \in C$; $\mathcal{A}^*$ also initializes $\mathcal{A}$ with $Mt_{\hat{S}_G,\hat{S}_M}$, $\hat{Mt}_{mult}$, $Mt_{\hat{S}_G,\hat{S}_H}$ and $Mt_{\hat{S}_H,\hat{S}_M}$ (as if $\mathcal{A}$ is the recommender in *X-Rec*);
3. $\mathcal{A}^*$ generates a random profile for every non-corrupted user, encrypts it under key $\hat{S}_M$ and sends it to $\mathcal{A}$;
4. $\mathcal{A}^*$ sends u's profile to F and receives u's prediction from F for every $u \in C$;
5. $\mathcal{A}^*$, on behalf of the service provider, receives an encryption w_u under $\hat{S}_M$ for every $u \in C$ from $\mathcal{A}$ (as if $\mathcal{A}$ is the recommender in *X-Rec*);
6. $\mathcal{A}^*$ decrypts w_u and calculates the randomness R from the decryption of w_u;
7. $\mathcal{A}^*$ masks R with the prediction for u (received from F), encrypts it under $\hat{S}_G$ and sends the encryption back to $\mathcal{A}$ (as if $\mathcal{A}$ is the user u in *X-Rec*);
8. $\mathcal{A}^*$ also runs the ideal comparison system with $\mathcal{A}$;
9. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

Note that $\mathcal{A}^*$ runs $\mathcal{A}$ as a black-box and is able to fix the random tape of $\mathcal{A}$. Thus $\mathcal{A}^*$ is able to determine the user sample which $\mathcal{A}$ uses in calculating the prediction for any user u. Moreover, $\mathcal{A}$ is honest-but-curious; thus since $\mathcal{A}^*$ knows all the user profiles which $\mathcal{A}$'s prediction algorithm is based on, $\mathcal{A}^*$ is able to calculate the prediction in the same way as $\mathcal{A}$ but in plaintext. As a result, $\mathcal{A}^*$ is able to calculate R from the decryption. Therefore, the transcript between $\mathcal{A}$ and $\mathcal{A}^*$ is computationally indistinguishable from the transcript between $\mathcal{A}$ and the non-corrupted users and the service provider by Lemma 3 and Lemma 4. Then the joint output of $\mathcal{A}$ and all other honest parties, and the joint output of $\mathcal{A}^*$ and all other honest parties are computationally indistinguishable.

Second, we consider the case where more than one users are corrupted and the service provider is also corrupted. For every algorithm $\mathcal{A}$ that corrupts a set of users C, and the service provider, we build an algorithm $\mathcal{A}^*$ that corrupts C and the service provider in the ideal recommendation system as follows.

1. $\mathcal{A}^*$ generates three keys: $\hat{S}_G$, $\hat{S}_M$ and $\hat{S}_H$, and three key-switching matrices: $Mt_{\hat{S}_G,\hat{S}_M}$, $\hat{Mt}_{mult}$, $Mt_{\hat{S}_G,\hat{S}_H}$ and $Mt_{\hat{S}_H,\hat{S}_M}$;
2. $\mathcal{A}^*$ initializes $\mathcal{A}$ with $Mt_{\hat{S}_G,\hat{S}_M}$ and $hatS_G$ for every corrupted user $u \in C$; $\mathcal{A}^*$ also initializes $\mathcal{A}$ with $Mt_{\hat{S}_M,\hat{S}_G}$, $\hat{S}_H$ (as if $\mathcal{A}$ is the service provider in *X-Rec*);

3. for every non-corrupted user, $\mathcal{A}^*$ sends $\mathcal{A}$ an encryption under key $\hat{S}_M$ of some random profile (for $\mathcal{A}$ to switch the key from $\hat{S}_M$ to $\hat{S}_G$);

4. for every corrupted user u, $\mathcal{A}^*$ sends u's profile to F and receives u's prediction from F;

5. for every corrupted user u, $\mathcal{A}^*$ encrypts u's prediction masked with a randomness R under key $\hat{S}_M$ to $\mathcal{A}$ (as if $\mathcal{A}$ is the service provider in *X-Rec*) and also sends R to $\mathcal{A}$ (as if $\mathcal{A}$ is the user u);

6. $\mathcal{A}^*$ also runs the ideal comparison system with $\mathcal{A}$;

7. $\mathcal{A}^*$ outputs whatever $\mathcal{A}$ outputs.

It is easy to see that the transcript between $\mathcal{A}$ and $\mathcal{A}^*$ is computationally indistinguishable from the transcript between $\mathcal{A}$ and the non-corrupted users and the service provider by Lemma 3 and Lemma 4. Then the joint output of $\mathcal{A}$ and all other honest parties, and the joint output of $\mathcal{A}^*$ and all other honest parties are computationally indistinguishable.

In both cases, the joint output is indistinguishable between the real recommendation and the ideal recommendation. Hence the online recommendation presented in Section 4 achieves the system-level privacy even if we allow collusion between parties.

5.3.2 User-level privacy

Recall the user-level privacy definition. Since more parties are corrupted, we need to define a new algorithm $\mathcal{B}_{u_1,u_2,\ldots,u_k}, k < N+1$ that involves the same n parties. $\mathcal{B}_{u_1,u_2,\ldots,u_k}, k < N+1$ works the same as Algorithm 1 except that $\mathcal{B}_{u_1,u_2,\ldots,u_k}$ only has outputs to $u_1, u_2, \ldots, u_k$.

Denote by d_{N+1} an input to $\mathcal{B}_{u_1,u_2,\ldots,u_k}$ that contains $N+1$ user profiles. Denote by d an input such that d_{N+1} contains exactly one more user profile than d. Let $X_{N+1} = \mathcal{B}_{u_1,u_2,\ldots,u_k}(d_{N+1})$ and $X = \mathcal{B}_{u_1,u_2,\ldots,u_k}(d)$. Then the proof of Lemma 6 also works here if we replace the definition of X and X_{N+1} in that proof. Therefore, for all d_{N+1}, d, $\mathcal{B}_{u_1,u_2,\ldots,u_k}(d_{N+1})$ and $\mathcal{B}_{u_1,u_2,\ldots,u_k}(d)$ are $(\ln \frac{1}{1-F}, F)$-indistinguishable for all u.

Then by the user-level privacy definition, then the prediction algorithm presented in Section 4 achieves the user-level privacy even if we allow collusion between users.

Now if we allow collusion between users and the recommender, an adversary $\mathcal{A}$ additionally knows which users are sampled. Theorem 3 does not apply here: the indistinguishability of whether a target user profile t is sampled or not is broken by $\mathcal{A}$'s additional knowledge.

We focus on how an adversary $\mathcal{A}$ may solve t with the help of this knowledge. To deduce the preference on item j of t, $\mathcal{A}$ needs $m \geq s$ predictions such that (1) only m users' data contribute to these m predictions; (2) t is among the m users (otherwise, what $\mathcal{A}$ collects imposes only constraints on t, which $\mathcal{A}$ does not know how to solve).

This turns to be an $m \times m$ equation system if $\mathcal{A}$ knows the sign of each similarity τ. Since in our adversarial model, $\mathcal{A}$ does not have the knowledge of other users, the best $\mathcal{A}$ can do is to guess the m signs of which the success probability is upper-bounded by $\frac{1}{2^s}$, which is negligible in X-Rec. As a result, the prediction algorithm presented in Section 4 achieves the user-level privacy even if we allow collusion between users and the recommender.

5.4 Privacy analysis of user profiles based on ideal systems and algorithms

Now that the recommendation, comparison protocol *X-NN* and prediction are indistinguishable from the ideal systems and algorithm, we could analyze the secrecy of user profiles by only looking at the ideal ones.

- **For parties that have no output.** Two parties have no output as defined by the functionality: the recommender and the service provider. In the ideal system, the service provider actually does not interact with the oracle and hence could learn nothing. The recommender learns user profiles encrypted under S_M and is equipped with an oracle which does bit decomposition over ciphertexts encrypted under S_H. However, by Lemma 4, the secret information which the recommender has:

  $$S_G, Mt_{S_M,S_H}, Mt_{S_H,S_M} \text{ and } Mt_{S_G,S_M}$$

  does not help the recommender in recovering S_M and S_H. Thus, by Lemma 3,

no polynomial-time recommender could learn things of user profiles.

- **For parties that have outputs.** All users (or their trusted proxies) have predicted ratings as outputs as defined by the functionality. In the ideal recommendation, users only receive predicted ratings. In other words, if users could learn something, users only learn it from predicted ratings. However, in the ideal prediction, any user u could do no interaction with the oracle, which disables the other users from learning u's profile.

6 Alternative adversaries

In this section, we will examine the impact of malicious parties and other possible attacks against a recommender system. (while in the other sections, we assume that all corrupted parties are honest-but-curious).

In particular, we show that *X-Rec* tolerates malicious recommender and malicious service provider to some extent but not malicious users; traffic analysis and access pattern are complementary to the adversary model in the report and countermeasures exist to mitigate their impact on *X-Rec*.

- **Malicious recommender**. A malicious recommender may modify, inject, or delete ciphertexts: encrypted user profiles and/or encrypted predicted ratings. Such recommender may even mix user profiles, substitute one user profile for another. A malicious recommender may also continuously request encrypted bit decomposition from the service provider. However, since the malicious recommender does not see the real ratings or bits, the recommender should not be able to make sense of those actions.

- **Malicious service provider**. A malicious service provider may modify, inject, or delete encrypted predicted ratings when they are sent back to users. The service provider may also respond incorrectly to the recommender's request for encrypted bit decomposition. Since the malicious service provider does not see the real ratings and the integer to be decomposed is masked with randomness, the service provider should not be able to make sense of those actions.

  The service provider may also mix those ratings, substitute predicted ratings to one user for others. However, this will be noticed by users. Since these ratings are masked with randomness, and if users cannot make sense of the ratings received, the corresponding ciphertexts must have been tampered.

- **Malicious users**. A malicious user may create many fake user profiles by creating many identities in the recommender system. This is called the Sybil attack [6]. *X-Rec* does not detect the Sybil attack. The resistance to such attack requires that some party (or parties) is able to distinguish between fake user profiles and genuine user profiles. However, *X-Rec* ensures the secrecy of user profiles by ensuring that none is able to distinguish encrypted user profiles, which is contradictory to the resistance to the Sybil attack.

  Unscrupulous users or compromised clients may submit artificial ratings in order to have certain products recommended more often than their competitors. This is called the Shilling attack [15]. Similarly, the resistance to the Shilling attack provides a method to distinguish between genuine ratings and artificial ones, which is contradictory to the privacy goal of *X-Rec*.

- **Traffic analysis**. An adversary Alice (who may or may not be users, the recommender, or the service provider) may listen to the network traffic of a target user with the recommender system. Alice may collect the packet departure/arrival time, message lengths, etc. to deduce user behavior.

  If Alice also corrupts the recommender or colludes with the recommender, she is able to link an identity in the recommender system with an IP address (where the package of the encrypted user profile comes from).

  Users can take countermeasures such as mix networks and Tor to mitigate the traffic analysis [8].

- **Access pattern**. To reduce latency, *X-Rec* does item sampling. I.e. instead of updating

a vector of ratings each time, the user proxy updates only the rating recently rated mixed with $L-1$ other ratings where L is a parameter of *X-Rec*. Since all ratings are encrypted, item sampling does not break the secrecy of user profiles.

If we consider the update of one rating as a write request on the database of the recommender, then the database can be seen as a write-only Oblivious Random Access Memory (ORAM) [12]. Unfortunately, the access pattern of such writes does not satisfy the security definition of ORAM [25]. In other words, the recommender may deduce information of users from the the updates of ratings. A possible countermeasure may be to mix movie items in the database frequently so that the recommender is unable to make sense of the updates.

References

[1] E. Aïmeur, G. Brassard, J. Fernandez, and F. Mani Onana. Alambic: a privacy-preserving recommender system for electronic commerce. *International Journal of Information Security*, 7(5), 2008.

[2] M. Albrecht, C. Cid, J.-C. Faugre, R. Fitzpatrick, and L. Perret. On the complexity of the bkw algorithm on lwe. *Designs, Codes and Cryptography*, 74(2), 2015.

[3] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `http://eprint.iacr.org/`.

[4] R. Bassily, A. Groce, J. Katz, and A. Smith. Coupled-worlds privacy: exploiting adversarial uncertainty in statistical data privacy. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, 2013.

[5] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1), 2000.

[6] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, 2002.

[7] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM (CACM)*, 54(1), Jan. 2011.

[8] M. Edman and B. Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys (CSUR)*, 42(1), Dec. 2009.

[9] A. Elmisery, S. Rho, and D. Botvich. Collaborative privacy framework for minimizing privacy risks in an iptv social recommender service. *Multimedia Tools and Applications*, 2014.

[10] Z. Erkin, M. Beye, T. Veugen, and R. Lagendijk. Efficiently computing private recommendations. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011.

[11] O. Goldreich. *Foundations of Cryptography*. Cambridge university press, 2004.

[12] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM (JACM)*, 43(3), May 1996.

[13] S. Guha, B. Cheng, and P. Francis. Privad: Practical privacy in online advertising. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, 2011.

[14] A. Jeckmans, A. Peter, and P. Hartel. Efficient privacy-enhanced familiarity-based recommender system. In J. Crampton, S. Jajodia, and K. Mayes, editors, *Computer Security - ESORICS 2013*, volume 8134 of *LNCS*. 2013.

[15] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, 2004.

[16] A. K. Lenstra. Key lengths. In *The Handbook of Information Security*. 2004.

[17] R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Topics in Cryptology - CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*. 2011.

[18] B. Liu and U. Hengartner. ptwitterrec: A privacy-preserving personalized tweet recommendation framework. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, 2014.

[19] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, 2009.

[20] D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-Quantum Cryptography 2009*.

[21] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, 2009.

[22] N. Ramakrishnan, B. Keller, B. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *Internet Computing, IEEE*, 5(6), Nov 2001.

[23] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, 2005.

[24] M. Rückert and M. Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137. `http://eprint.iacr.org/`.

[25] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, 2013.

[26] H. Zhou and G. Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *Information Theory and Applications Workshop (ITA), 2014*, 2014.