Privacy-preserving collaborative filtering for the cloud

Anirban Basu\(^1\)  Jaideep Vaidya\(^2\)  Hiroaki Kikuchi\(^1\)
Theo Dimitrakos\(^3\)

\(^1\)Graduate School of Engineering, Tokai University, Japan
\(^2\)MSIS Department, Rutgers The State University of New Jersey, USA
\(^3\)Research & Technology, British Telecom, UK

IEEE Cloudcom 2011, Athens, Greece
Outline

1. Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2. Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3. Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4. Evaluation
   - Implementation and results

5. Tailpiece
   - Conclusions and future work
   - Question time!
A recommendation example: Amazon’s “people who buy this also buy that” (user profile analysis).

Rating-based collaborative filtering (CF) – another mechanism for recommendation.
A recommendation example: Amazon’s “people who buy this also buy that” (user profile analysis).

Rating-based collaborative filtering (CF) – another mechanism for recommendation.
## Recommendation and CF

- **Items**: $i_1, i_2, i_3, \ldots, i_k, \ldots, i_n$
- **Users**: $u_1, u_2, \ldots, u_m$

### Sparse user-item rating matrix

$$\begin{array}{cccc}
\text{Users} & u_1 & u_2 & \ldots & u_m \\
\text{Items} & i_1 & i_2 & i_3 & \ldots & i_k & \ldots & i_n \\
\text{Predict:} & u_x & & & \text{?} & & & i_k \\
\end{array}$$

The task is to predict the rating user $u_x$ will give to item $i_k$ given the sparse user-item rating matrix.
An airlines example ("-" implies absence of ratings):

<table>
<thead>
<tr>
<th></th>
<th>Virgin Atlantic</th>
<th>Emirates</th>
<th>Singapore Airlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>3</td>
<td>?</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Tracy</td>
<td>-</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Steve</td>
<td>3</td>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

**Predict: how would Alice rate Emirates?**
CF on the cloud – privacy risks

- Recommendation providers may run on cloud computing infrastructures.
- Your private rating data may not be safe on the cloud because of insider and outsider threats.
CF on the cloud – privacy risks

- Recommendation providers may run on cloud computing infrastructures.
- Your private rating data may not be safe on the cloud because of insider and outsider threats.
These are where privacy concerns are raised.
Research problem: privacy preserving CF

Compute a privacy preserving rating prediction on a Software-as-a-Service (SaaS) construction Platform-as-a-Service (PaaS) cloud, such that we are:

- able to hide and/or delink user’s private rating data without using any trusted third party,
- and it is robust to insider threats from the cloud,
- while assuming honest-but-curious user,
- and assuming identity concealing network infrastructures (e.g. anonymous networks, pseudonyms, IPv4 NAT).
The research problem

Research problem: privacy preserving CF

Compute a privacy preserving rating prediction on a Software-as-a-Service (SaaS) construction Platform-as-a-Service (PaaS) cloud, such that we are:

- able to hide and/or delink user’s private rating data without using any trusted third party,
- and it is robust to insider threats from the cloud,
- while assuming honest-but-curious user,
- and assuming identity concealing network infrastructures (e.g. anonymous networks, pseudonyms, IPv4 NAT).
The research problem

Research problem: privacy preserving CF

Compute a privacy preserving rating prediction on a Software-as-a-Service (SaaS) construction Platform-as-a-Service (PaaS) cloud, such that we are:

- able to hide and/or delink user’s private rating data without using any trusted third party,
- and it is robust to insider threats from the cloud,
- **while assuming honest-but-curious user,**
- and assuming identity concealing network infrastructures (e.g. anonymous networks, pseudonyms, IPv4 NAT).
Research problem: privacy preserving CF

Compute a privacy preserving rating prediction on a Software-as-a-Service (SaaS) construction Platform-as-a-Service (PaaS) cloud, such that we are:

- able to hide and/or delink user’s private rating data without using any trusted third party,
- and it is robust to insider threats from the cloud,
- while assuming honest-but-curious user,
- and assuming identity concealing network infrastructures (e.g. anonymous networks, pseudonyms, IPv4 NAT).
Our contributions

- A privacy preserving CF solution for the Google App Engine for Java (GAE/J)$^1$ – a specialised SaaS construction PaaS cloud.
- Can be extended to vertical partitions$^2$.
- Feasible on a real world public PaaS cloud.

$^1$http://code.google.com/appengine/

$^2$See § IV.C in the paper. Left out of this presentation for simplicity.
Our contributions

- A privacy preserving CF solution for the Google App Engine for Java (GAE/J)\(^1\) – a specialised SaaS construction PaaS cloud.

- Can be extended to vertical partitions\(^2\).

- Feasible on a real world public PaaS cloud.

---

\(^{1}\)http://code.google.com/appengine/

\(^{2}\)See § IV.C in the paper. Left out of this presentation for simplicity.
Our contributions

- A privacy preserving CF solution for the Google App Engine for Java (GAE/J)\(^1\) – a specialised SaaS construction PaaS cloud.
- Can be extended to vertical partitions\(^2\).
- Feasible on a real world public PaaS cloud.

\(^1\)http://code.google.com/appengine/
\(^2\)See § IV.C in the paper. Left out of this presentation for simplicity.
Outline

1. Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2. Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3. Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4. Evaluation
   - Implementation and results

5. Tailpiece
   - Conclusions and future work
   - Question time!
Types of collaborative filtering

CF can be:
- either memory based using similarity or deviations between users (user-based) or items (item-based);
- or model based, such as utilising the singular value decomposition technique.
Types of collaborative filtering

CF can be:

- either memory based using similarity or deviations between users (user-based) or items (item-based);
- or model based, such as utilising the singular value decomposition technique.
Privacy preserving CF – the state-of-the-art

Privacy-preserving CF

Classified, as per mechanism:

- **Encryption based** – where privacy of data is preserved through homomorphic encryption [Canny2002a, Han2009].

- **Randomisation based** – where privacy of data is preserved through random perturbation of the data or by anonymising identities [Polat2003, 2005 and 2006].
Privacy-preserving CF

Classified, as per mechanism:

- Encryption based – where privacy of data is preserved through homomorphic encryption [Canny2002a, Han2009].
- Randomisation based – where privacy of data is preserved through random perturbation of the data or by anonymising identities [Polat2003, 2005 and 2006].
Privacy-preserving CF

Classified, as per mechanism:

- Encryption based – where privacy of data is preserved through homomorphic encryption [Canny2002a, Han2009].
- Randomisation based – where privacy of data is preserved through random perturbation of the data or by anonymising identities [Polat2003, 2005 and 2006].

Classified, as per infrastructure, PPCF can be:

- single machine or single cluster based [Tada2010, Basu2011], or
- large-scale distributed [Berkovsky2007, Canny2002b].
Privacy-preserving CF

Classified, as per mechanism:

- Encryption based – where privacy of data is preserved through homomorphous encryption [Canny2002a, Han2009].
- Randomisation based – where privacy of data is preserved through random perturbation of the data or by anonymising identities [Polat2003, 2005 and 2006].

Classified, as per infrastructure, PPCF can be:

- single machine or single cluster based [Tada2010, Basu2011], or
- large-scale distributed [Berkovsky2007, Canny2002b].
I will not bore you with bibliography slides at the end... 

Please see the the paper for detailed references of the cited work.
Outline

1. Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2. Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3. Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4. Evaluation
   - Implementation and results

5. Tailpiece
   - Conclusions and future work
   - Question time!
What is Slope One?

The original paper on SlopeOne CF:

Collaborative filtering (CF) predictors of the form $f(x) = x + b$, hence “slope one”.

- Weighted version is based on pre-computed average deviations between ratings of items, weighted by relative cardinalities of pairs of items.
- Accurate, fast and incrementally updatable.
What is Slope One?

- Collaborative filtering (CF) predictors of the form $f(x) = x + b$, hence “slope one”.

- Weighted version is based on pre-computed average deviations between ratings of items, weighted by relative cardinalities of pairs of items.

  Accurate, fast and incrementally updatable.
Collaborative filtering (CF) predictors of the form
\[ f(x) = x + b, \] hence “slope one”.

Weighted version is based on pre-computed average deviations between ratings of items, weighted by relative cardinalities of pairs of items.

Accurate, fast and incrementally updatable.
Why Slope One?

The choice of the CF scheme has effect on performance and privacy on the cloud.

- **Traditional user-based or item-based CF requires **storage of private rating data**; easy to update but **slow to query**.

- Low-rank matrix approximations (e.g. SVD) are **difficult to compute incrementally**; otherwise **slow to update from stored private rating data** but fast to query.

- Slope One uses an incrementally updatable item-item matrix model; fast to update and fast to query.
Why Slope One?

The choice of the CF scheme has effect on performance and privacy on the cloud.

- Traditional user-based or item-based CF requires **storage of private rating data**; easy to update but **slow to query**.
- Low-rank matrix approximations (e.g. SVD) are **difficult to compute incrementally**; otherwise **slow to update** from stored private rating data but fast to query.
- Slope One uses an incrementally updatable item-item matrix model; fast to update and fast to query.
Why Slope One?

The choice of the CF scheme has effect on performance and privacy on the cloud.

- Traditional user-based or item-based CF requires **storage of private rating data**; easy to update but **slow to query**.
- Low-rank matrix approximations (e.g. SVD) are **difficult to compute incrementally**; otherwise **slow to update** from stored private rating data but fast to query.
- Slope One uses an incrementally updatable item-item matrix model; fast to update and fast to query.
Collaborative filtering and privacy
- Recommendation through collaborative filtering
- Collaborative filtering on the cloud and privacy
- The research problem
- Our contributions

Related work and background
- Privacy preserving CF – the state-of-the-art
- Slope One – a collaborative filtering predictor
- The generalised weighted Slope One

Proposed scheme
- Privacy-preserving CF
- Piecing it together

Evaluation
- Implementation and results

Tailpiece
- Conclusions and future work
- Question time!
The generalised weighted Slope One

The weighted Slope One

- The average deviations of ratings from item $a$ to item $b$ is given as:

\[
\delta_{a,b} = \frac{\Delta_{a,b}}{\phi_{a,b}} = \frac{\sum_i \delta_{i,a,b}}{\phi_{a,b}} = \frac{\sum_i (r_{i,a} - r_{i,b})}{\phi_{a,b}}
\]

where $\phi_{a,b}$ is the count of the users who have rated both items while $\delta_{i,a,b} = r_{i,a} - r_{i,b}$ is the deviation of the rating of item $a$ from that of item $b$ both given by user $i$.

- Thus, the rating for user $u$ and item $x$ using the weighted Slope One is predicted as:

\[
r_{u,x} = \frac{\sum_{a \neq x} (\delta_{x,a} + r_{u,a}) \phi_{x,a}}{\sum_{a \neq x} \phi_{x,a}} = \frac{\sum_{a \neq x} (\Delta_{x,a} + r_{u,a} \phi_{x,a})}{\sum_{a \neq x} \phi_{x,a}}
\]
The generalised weighted Slope One

The weighted Slope One

The average deviations of ratings from item $a$ to item $b$ is given as:

$$
\delta_{a,b} = \frac{\Delta_{a,b}}{\phi_{a,b}} = \frac{\sum_i \delta_{i,a,b}}{\phi_{a,b}} = \frac{\sum_i (r_{i,a} - r_{i,b})}{\phi_{a,b}}
$$  \hspace{1cm} (1)

where $\phi_{a,b}$ is the count of the users who have rated both items while $\delta_{i,a,b} = r_{i,a} - r_{i,b}$ is the deviation of the rating of item $a$ from that of item $b$ both given by user $i$.

Thus, the rating for user $u$ and item $x$ using the weighted Slope One is predicted as:

$$
r_{u,x} = \frac{\sum_{a \neq x} (\delta_{x,a} + r_{u,a})\phi_{x,a}}{\sum_{a \neq x} \phi_{x,a}} = \frac{\sum_{a \neq x} (\Delta_{x,a} + r_{u,a}\phi_{x,a})}{\sum_{a \neq x} \phi_{x,a}}
$$  \hspace{1cm} (2)
Pre-computed incrementally updatable matrices

Weighted Slope One predictor has the following two pre-computed, incrementally updatable matrices.

- **Deviation matrix** or $\Delta$: each element is the total deviation of ratings between a pair of items, calculated over cases where both items have been rated by the same user. If the ratings matrix is of dimension $m \times n$ (i.e. $n$ items) then $\Delta$ is of dimension $n \times n$.

- **Cardinality matrix** or $\phi$: each element is the count of the cases where items in a pair have been both rated by the same user. It is of the same dimension as $\Delta$. 
The generalised weighted Slope One

**Pre-computed incrementally updatable matrices**

Weighted Slope One predictor has the following two pre-computed, incrementally updatable matrices.

- **Deviation matrix** or \( \Delta \): each element is the total deviation of ratings between a pair of items, calculated over cases where both items have been rated by the same user. If the ratings matrix is of dimension \( mxn \) (i.e. \( n \) items) then \( \Delta \) is of dimension \( nxn \).

- **Cardinality matrix** or \( \phi \): each element is the count of the cases where items in a pair have been both rated by the same user. It is of the same dimension as \( \Delta \).
Pre-computed incrementally updatable matrices

The generalised weighted Slope One

Slope One pre-computation phase

Sparse user-item rating matrix $(m \times n)$

Sparse item-item deviation and cardinality matrices $(n \times n)$

indicates private data

Anirban Basu, et al.
Cloud based privacy preserving CF
Outline

1. Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2. Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3. Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4. Evaluation
   - Implementation and results

5. Tailpiece
   - Conclusions and future work
   - Question time!
Additively homomorphic Paillier cryptosystem

- homomorphic addition:
  \[ E(m_1 + m_2) = E(m_1) \cdot E(m_2) \]

- homomorphic multiplication:
  \[ E(m_1 \cdot \pi) = E(m_1)^\pi \]

We denote encryption and decryption functions as \( E() \) and \( D() \) respectively with plaintext messages \( m_1, m_2 \) and integer multiplicand \( \pi \).
Additively homomorphic Paillier cryptosystem

- Homomorphic addition:

\[ \mathcal{E}(m_1 + m_2) = \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) \]

- Homomorphic multiplication:

\[ \mathcal{E}(m_1 \cdot \pi) = \mathcal{E}(m_1)^\pi \]

We denote encryption and decryption functions as \( \mathcal{E}(\cdot) \) and \( \mathcal{D}(\cdot) \) respectively with plaintext messages \( m_1, m_2 \) and integer multiplicand \( \pi \).
Based on the previous equation for plaintext Slope One predictors, we can write:

$$\sum_{a|a \neq x} (\Delta x, a + r_u, a \phi x, a) = D(\prod_{a|a \neq x} (E(\Delta x, a)(E(r_u, a)^{\phi x, a}))) \quad (3)$$

and reducing the number of encryptions, the final prediction is given as:

$$r_{u,x} = \frac{D(E(\sum_{a|a \neq x} \Delta x, a) \prod_{a|a \neq x} (E(r_u, a)^{\phi x, a}))}{\sum_{a|a \neq x} \phi x, a} \quad (4)$$
Based on the previous equation for plaintext Slope One predictors, we can write:

\[
\sum_{a \neq x} (\Delta_{x,a} + r_{u,a}\phi_{x,a}) = D\left( \prod_{a \neq x} (E(\Delta_{x,a})(E(r_{u,a})^{\phi_{x,a}})) \right) \tag{3}
\]

and reducing the number of encryptions, the final prediction is given as:

\[
r_{u,x} = \frac{D(E(\sum_{a \neq x} \Delta_{x,a}) \prod_{a \neq x} (E(r_{u,a})^{\phi_{x,a}}))}{\sum_{a \neq x} \phi_{x,a}} \tag{4}
\]
Privacy-preserving Slope One

- Since $\Delta$ and $\phi$ are not private information with respect to user data, these are stored unencrypted in the cloud.
- These matrices are updated as ratings of items are added, updated or deleted in pairs.
- Proposed solution uses user-encrypted prediction query and response.
Privacy-preserving Slope One

- Since $\Delta$ and $\phi$ are not private information with respect to user data, these are stored unencrypted in the cloud.
- These matrices are updated as ratings of items are added, updated or deleted in pairs.
- Proposed solution uses user-encrypted prediction query and response.
Since $\Delta$ and $\phi$ are not private information with respect to user data, these are stored unencrypted in the cloud.

These matrices are updated as ratings of items are added, updated or deleted in pairs.

Proposed solution uses **user-encrypted** prediction query and response.
Outline

1. Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2. Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3. Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4. Evaluation
   - Implementation and results

5. Tailpiece
   - Conclusions and future work
   - Question time!
Overview of the proposed scheme

User submits plaintext pair-wise ratings or deviations of ratings.

Queries with encrypted (user’s public key) rating vector.

Returns encrypted prediction which only the user can decrypt.

Identity anonymiser

Google App Engine (GAE/J) or other PaaS cloud distributed datastore

CF application cloud app instance

Stores plaintext deviations and cardinalities.

Computes encrypted prediction from stored data.

CF application cloud app instance

PaaS cloud
De-linking identities with IPv4 NAT

A simple IPv4 NAT can provide a naïve approach to make linkability between actual users and their WAN side IPs hard.
Addition, update, deletion and prediction of ratings

Figure: UML sequence diagram for addition, update or deletion of data between any one user and the cloud-based CF site.

See algorithms IV.1-IV.3 in the paper.
Addition, update, deletion and prediction of ratings

Figure: UML sequence diagram for prediction of between any one user and the cloud-based CF site.

^3See algorithms IV.1-IV.3 in the paper.
Outline

1 Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2 Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3 Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4 Evaluation
   - Implementation and results

5 Tailpiece
   - Conclusions and future work
   - Question time!
Google App Engine for Java (GAE/J)

- Specialised SaaS construction PaaS cloud.
- SaaS application instances run on Java Virtual Machines with web front-ends.
- Automatically allocated scalable resources for growing user requests.
- Slow but high replication datastore access; and fast distributed in-memory cache.
Google App Engine for Java (GAE/J)

- Specialised SaaS construction PaaS cloud.
- **SaaS application instances run on Java Virtual Machines with web front-ends.**
  - Automatically allocated scalable resources for growing user requests.
  - Slow but high replication datastore access; and fast distributed in-memory cache.
Google App Engine for Java (GAE/J)

- Specialised SaaS construction PaaS cloud.
- SaaS application instances run on Java Virtual Machines with web front-ends.
- Automatically allocated scalable resources for growing user requests.
- Slow but high replication datastore access; and fast distributed in-memory cache.
Google App Engine for Java (GAE/J)

- Specialised SaaS construction PaaS cloud.
- SaaS application instances run on Java Virtual Machines with web front-ends.
- Automatically allocated scalable resources for growing user requests.
- Slow but high replication datastore access; and fast distributed in-memory cache.
Google App Engine for Java (GAE/J)

- Low CPU performance per application instance: affects cryptographic operations.
- Various performance limitations on the free quota.
- As of July 22, 2011, we measured some limitations of the GAE/J.
Google App Engine for Java (GAE/J)

- Low CPU performance per application instance: affects cryptographic operations.
- Various performance limitations on the free quota.
  - As of July 22, 2011, we measured some limitations of the GAE/J.
Google App Engine for Java (GAE/J)

- Low CPU performance per application instance: affects cryptographic operations.
- Various performance limitations on the free quota.
- As of July 22, 2011, we measured some limitations of the GAE/J.

Feasibility of a PPCF scheme on the GAE/J

Google App Engine for Java (GAE/J)

- Low CPU performance per application instance: affects cryptographic operations.
- Various performance limitations on the free quota.
- As of July 22, 2011, we measured some limitations of the GAE/J.

Feasibility of a PPCF scheme on the GAE/J

### Performance results on the GAE/J

<table>
<thead>
<tr>
<th>Bit size&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Vector size&lt;sup&gt;b&lt;/sup&gt;</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>5</td>
<td>500ms</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
<td>650ms</td>
</tr>
<tr>
<td>2048</td>
<td>5</td>
<td>3800ms</td>
</tr>
<tr>
<td>2048</td>
<td>10</td>
<td>5000ms</td>
</tr>
</tbody>
</table>

<sup>a</sup>Paillier cryptosystem modulus bit size, i.e. $|n|$.

<sup>b</sup>Size of the encrypted rating query vector.
Performance results on the GAE/J

Time taken to predict grows linearly . . .

. . . with the size of the query vector. With 100 given ratings in the query vector, the prediction time will be about 50 seconds – an awfully long wait on a web interface!

<table>
<thead>
<tr>
<th>Bit size(^a)</th>
<th>Vector size(^b)</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>5</td>
<td>500ms</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
<td>650ms</td>
</tr>
<tr>
<td>2048</td>
<td>5</td>
<td>3800ms</td>
</tr>
<tr>
<td>2048</td>
<td>10</td>
<td>5000ms</td>
</tr>
</tbody>
</table>

\(^a\)Paillier cryptosystem modulus bit size, i.e. \(|n|\).
\(^b\)Size of the encrypted rating query vector.
Demo


- Attack simulation on private data: in both cases, the cloud application tracks user’s IPv4 address – a typical attack scenario to attempt to link ratings to users.
Demo

- **Google App Engine for Java implementation:**
  
  http://gaejppcf.appspot.com/.

- **Attack simulation on private data:** In both cases, the cloud application tracks user’s IPv4 address – a typical attack scenario to attempt to link ratings to users.
Outline

1. Collaborative filtering and privacy
   - Recommendation through collaborative filtering
   - Collaborative filtering on the cloud and privacy
   - The research problem
   - Our contributions

2. Related work and background
   - Privacy preserving CF – the state-of-the-art
   - Slope One – a collaborative filtering predictor
   - The generalised weighted Slope One

3. Proposed scheme
   - Privacy-preserving CF
   - Piecing it together

4. Evaluation
   - Implementation and results

5. Tailpiece
   - Conclusions and future work
   - Question time!
Conclusions

Our proposed scheme:

- uses user-encrypted predicted query and does not store users’ rating data;
- makes rating-to-user linkability hard; and
- scales well on real world cloud platforms.
Conclusions

Our proposed scheme:

- uses user-encrypted predicted query and does not store users’ rating data;
- makes rating-to-user linkability hard; and
- scales well on real world cloud platforms.
Conclusions

Our proposed scheme:
- uses user-encrypted predicted query and does not store users’ rating data;
- makes rating-to-user linkability hard; and
- scales well on real world cloud platforms.
Future work

- Implement the proposal on vertical partition.
- Introduce parallelism in prediction queries with large query vectors.
- Conduct comparative performance analyses with other privacy preserving CF implementations on different SaaS construction PaaS clouds, e.g. the Amazon Elastic Beanstalk.
- Improve our scheme by discarding some assumptions (e.g. honest user) and dependencies (e.g. anonymiser networks).
Future work

- Implement the proposal on vertical partition.
- Introduce parallelism in prediction queries with large query vectors.
- Conduct comparative performance analyses with other privacy preserving CF implementations on different SaaS construction PaaS clouds, e.g. the Amazon Elastic Beanstalk.
- Improve our scheme by discarding some assumptions (e.g. honest user) and dependencies (e.g. anonymiser networks).
**Future work**

- Implement the proposal on vertical partition.
- Introduce parallelism in prediction queries with large query vectors.
- Conduct comparative performance analyses with other privacy preserving CF implementations on different SaaS construction PaaS clouds, e.g. the Amazon Elastic Beanstalk.
- Improve our scheme by discarding some assumptions (e.g. honest user) and dependencies (e.g. anonymiser networks).
Future work

- Implement the proposal on vertical partition.
- Introduce parallelism in prediction queries with large query vectors.
- Conduct comparative performance analyses with other privacy preserving CF implementations on different SaaS construction PaaS clouds, e.g. the Amazon Elastic Beanstalk.
- Improve our scheme by discarding some assumptions (e.g. honest user) and dependencies (e.g. anonymiser networks).
Any questions?