

Practical privacy preserving collaborative filtering on the Google App Engine

Anirban Basu¹ Jaideep Vaidya² Hiroaki Kikuchi¹
Theo Dimitrakos³

¹Graduate School of Engineering, Tokai University, Japan

²MSIS Department, Rutgers The State University of New Jersey, USA

³Research & Technology, British Telecom, UK

CSS 2011, Niigata, Japan

Table of Contents

- 1 Overview
 - What are we trying to do?
 - The research problem
- 2 Building blocks
 - Slope One
 - The generalised weighted Slope One
- 3 Proposed scheme
 - Overview
 - Privacy-preserving Slope One
- 4 Evaluation
 - Implementation results
- 5 Conclusions and future work
- 6 Question time!

What are we trying to do?

Recommendation

We often have user-item rating data similar to this:

	Virgin Atlantic	Emirates	Singapore Airlines
Alice	3	?	5
Bob	3	4	5
Tracy	-	2	4
Steve	3	3	-

- The objective is to recommend the rating for Emirates to Alice using collaborative filtering (CF).
- CF can be either memory based using similarity or deviations between users (user-based) or items (item-based); or be model based, such as the singular value decomposition.

What are we trying to do?

Recommendation

We often have user-item rating data similar to this:

	Virgin Atlantic	Emirates	Singapore Airlines
Alice	3	?	5
Bob	3	4	5
Tracy	-	2	4
Steve	3	3	-

- The objective is to recommend the rating for Emirates to Alice using collaborative filtering (CF).
- CF can be either memory based using similarity or deviations between users (user-based) or items (item-based); or be model based, such as the singular value decomposition.

What are we trying to do?

Recommendation on the cloud

- **A recommendation example: Amazon's "people who buy x also buy y ".**
- Recommendation providers (e.g. Amazon, eBay) run on cloud computing infrastructures.
- But your private rating data is not safe on the cloud because of:

What are we trying to do?

Recommendation on the cloud

- A recommendation example: Amazon's "people who buy x also buy y ".
- Recommendation providers (e.g. Amazon, eBay) run on cloud computing infrastructures.
- But your private rating data is not safe on the cloud because of:

What are we trying to do?

Recommendation on the cloud

- A recommendation example: Amazon's "people who buy x also buy y ".
- Recommendation providers (e.g. Amazon, eBay) run on cloud computing infrastructures.
- **But your private rating data is not safe on the cloud because of:**
 - insider threats of the provider or the cloud computing infrastructure, and
 - outsider threats from attacks.

What are we trying to do?

Recommendation on the cloud

- A recommendation example: Amazon's "people who buy x also buy y ".
- Recommendation providers (e.g. Amazon, eBay) run on cloud computing infrastructures.
- But your private rating data is not safe on the cloud because of:
 - insider threats of the provider or the cloud computing infrastructure, and
 - outsider threats from attacks.

What are we trying to do?

Recommendation on the cloud

- A recommendation example: Amazon's "people who buy x also buy y ".
- Recommendation providers (e.g. Amazon, eBay) run on cloud computing infrastructures.
- But your private rating data is not safe on the cloud because of:
 - insider threats of the provider or the cloud computing infrastructure, and
 - **outsider threats from attacks.**

Research problem: privacy preserving CF

Compute a rating prediction through privacy preserving CF on a Platform-as-a-Service (PaaS) cloud. Requirements are:

- to hide user's private rating data without depending any trusted third party for threshold decryption,
- to assume honest user,
- to expect insider threats from the cloud,
- to assume identity concealing network configurations (e.g. anonymous networks, pseudonyms, IPv4 NAT).

Research problem: privacy preserving CF

Compute a rating prediction through privacy preserving CF on a Platform-as-a-Service (PaaS) cloud. Requirements are:

- to hide user's private rating data without depending any trusted third party for threshold decryption,
- **to assume honest user,**
- to expect insider threats from the cloud,
- to assume identity concealing network configurations (e.g. anonymous networks, pseudonyms, IPv4 NAT).

Research problem: privacy preserving CF

Compute a rating prediction through privacy preserving CF on a Platform-as-a-Service (PaaS) cloud. Requirements are:

- to hide user's private rating data without depending any trusted third party for threshold decryption,
- to assume honest user,
- **to expect insider threats from the cloud,**
- to assume identity concealing network configurations (e.g. anonymous networks, pseudonyms, IPv4 NAT).

Research problem: privacy preserving CF

Compute a rating prediction through privacy preserving CF on a Platform-as-a-Service (PaaS) cloud. Requirements are:

- to hide user's private rating data without depending any trusted third party for threshold decryption,
- to assume honest user,
- to expect insider threats from the cloud,
- to assume identity concealing network configurations (e.g. anonymous networks, pseudonyms, IPv4 NAT).

Research problem: privacy preserving CF

Can we predict ratings for the user from some incrementally pre-computed compact model instead of users' private rating data?

- **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; slow to update but fast to query.
- Slope One (I will explain soon!) uses an incrementally updatable item-item matrix model; fast to update and fast to query.

Research problem: privacy preserving CF

Can we predict ratings for the user from some incrementally pre-computed compact model instead of users' private rating data?

- 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; slow to update but fast to query.**
- Slope One (I will explain soon!) uses an incrementally updatable item-item matrix model; fast to update and fast to query.

Research problem: privacy preserving CF

Can we predict ratings for the user from some incrementally pre-computed compact model instead of users' private rating data?

- 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; slow to update but fast to query.
- **Slope One (I will explain soon!) uses an incrementally updatable item-item matrix model; fast to update and fast to query.**

The research problem

Our contributions

- We propose a privacy preserving SlopeOne CF on the Google App Engine for Java (GAE/J)¹ – a specialised SaaS construction PaaS cloud.
- Proposed scheme can be extended to vertical partitions.
- Feasible on a real world public cloud; also tested with Amazon Web Services Elastic Beanstalk (AWS EBS)² PaaS cloud.

¹<http://code.google.com/appengine/>

²<http://aws.amazon.com/elasticbeanstalk/>

The research problem

Our contributions

- We propose a privacy preserving SlopeOne CF on the Google App Engine for Java (GAE/J)¹ – a specialised SaaS construction PaaS cloud.
- **Proposed scheme can be extended to vertical partitions.**
- Feasible on a real world public cloud; also tested with Amazon Web Services Elastic Beanstalk (AWS EBS)² PaaS cloud.

¹<http://code.google.com/appengine/>

²<http://aws.amazon.com/elasticbeanstalk/>

The research problem

Our contributions

- We propose a privacy preserving SlopeOne CF on the Google App Engine for Java (GAE/J)¹ – a specialised SaaS construction PaaS cloud.
- Proposed scheme can be extended to vertical partitions.

The extension of our work

A. Basu, J. Vaidya, H. Kikuchi and T. Dimitrakos,
Privacy-preserving collaborative filtering for the cloud,
Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (IEEE Cloudcom), Athens, Greece, 2011.

- Feasible on a real world public cloud; also tested with Amazon Web Services Elastic Beanstalk (AWS EBS)² PaaS cloud.

¹<http://code.google.com/appengine/>

Our contributions

- We propose a privacy preserving SlopeOne CF on the Google App Engine for Java (GAE/J)¹ – a specialised SaaS construction PaaS cloud.
- Proposed scheme can be extended to vertical partitions.
- Feasible on a real world public cloud; also tested with Amazon Web Services Elastic Beanstalk (AWS EBS)² PaaS cloud.

¹<http://code.google.com/appengine/>

²<http://aws.amazon.com/elasticbeanstalk/>

What is Slope One?

The original paper on SlopeOne CF:

Lemire, D., Maclachlan, A. 2005. *Slope one predictors for online rating-based collaborative filtering*. In: Society for Industrial Mathematics.

- Item-based collaborative filtering (CF) scheme 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 easy to update.

What is Slope One?

- Item-based collaborative filtering (CF) scheme 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 easy to update.

What is Slope One?

- Item-based collaborative filtering (CF) scheme 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 easy to update.

What is Slope One?

- Item-based collaborative filtering (CF) scheme 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 easy to update.**

Generalising the rationale

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

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

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:

$$r_{u,x} = \frac{\sum_{a|a \neq x} (\overline{\delta_{x,a}} + r_{u,a}) \phi_{x,a}}{\sum_{a|a \neq x} \phi_{x,a}} = \frac{\sum_{a|a \neq x} (\Delta_{x,a} + r_{u,a} \phi_{x,a})}{\sum_{a|a \neq x} \phi_{x,a}} \quad (2)$$

Generalising the rationale

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

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

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:

$$r_{u,x} = \frac{\sum_{a|a \neq x} (\overline{\delta_{x,a}} + r_{u,a}) \phi_{x,a}}{\sum_{a|a \neq x} \phi_{x,a}} = \frac{\sum_{a|a \neq x} (\Delta_{x,a} + r_{u,a} \phi_{x,a})}{\sum_{a|a \neq x} \phi_{x,a}} \quad (2)$$

Generalising the rationale

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

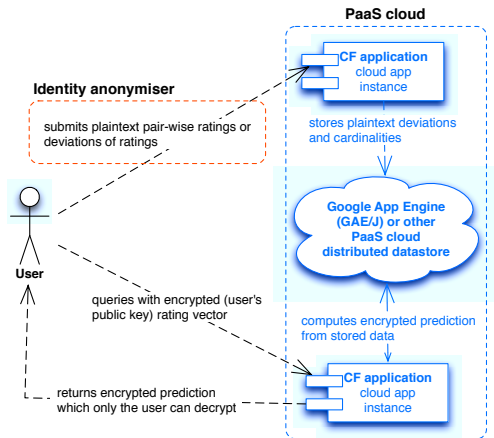
- **Deviation matrix** or Δ : each element is the total deviation of ratings between a pair of items, calculated over cases where both items have been rated by the same user. If the ratings matrix is of dimension $m \times n$ (i.e. n items) then Δ is of dimension $n \times n$.
- **Cardinality matrix**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

Generalising the rationale

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

- **Deviation matrix** or Δ : 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 Δ is of dimension $n \times n$.
- **Cardinality matrix** or ϕ : 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 Δ .

Proposed scheme using Slope One³



³See algorithms 3.1-3.4 in the paper.

Proposed scheme using Slope One³

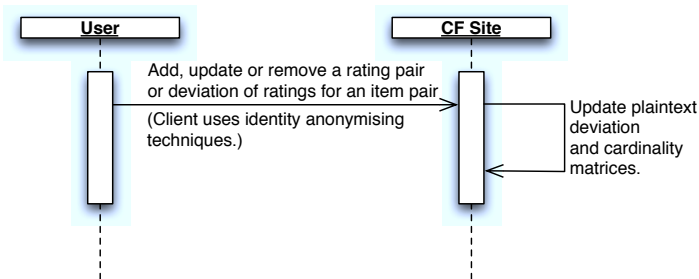


Figure: UML sequence diagram for addition, update or deletion of data between any one user and the cloud-based CF site.

³See algorithms 3.1-3.4 in the paper.

Proposed scheme using Slope One³

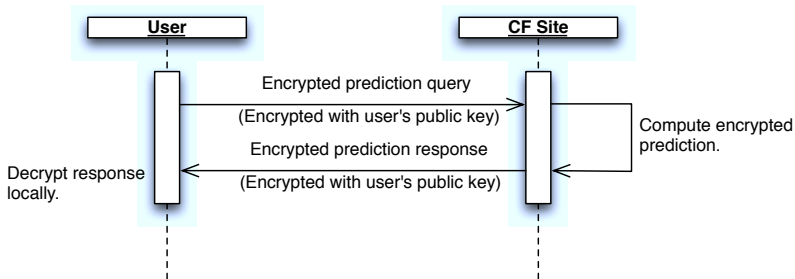


Figure: UML sequence diagram for prediction of between any one user and the cloud-based CF site.

³See algorithms 3.1-3.4 in the paper.

Privacy preserving Slope One CF

Additively homomorphic cryptosystem (Paillier) supports:

- **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 $\mathcal{E}()$ and $\mathcal{D}()$ respectively with plaintext messages m_1 , m_2 and integer multiplicand π .

Privacy preserving Slope One CF

Additively homomorphic cryptosystem (Paillier) supports:

- 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 $\mathcal{E}()$ and $\mathcal{D}()$ respectively with plaintext messages m_1 , m_2 and integer multiplicand π .

Privacy preserving Slope One CF

Based on the previous equation for plaintext Slope One predictors, we can write:

$$\sum_{a|a \neq x} (\Delta_{x,a} + r_{u,a} \phi_{x,a}) = \mathcal{D} \left(\prod_{a|a \neq x} (\mathcal{E}(\Delta_{x,a}) (\mathcal{E}(r_{u,a})^{\phi_{x,a}})) \right) \quad (3)$$

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

$$r_{u,x} = \frac{\mathcal{D}(\mathcal{E}(\sum_{a|a \neq x} \Delta_{x,a}) \prod_{a|a \neq x} (\mathcal{E}(r_{u,a})^{\phi_{x,a}}))}{\sum_{a|a \neq x} \phi_{x,a}} \quad (4)$$

Privacy preserving Slope One CF

Based on the previous equation for plaintext Slope One predictors, we can write:

$$\sum_{a|a \neq x} (\Delta_{x,a} + r_{u,a} \phi_{x,a}) = \mathcal{D} \left(\prod_{a|a \neq x} (\mathcal{E}(\Delta_{x,a}) (\mathcal{E}(r_{u,a})^{\phi_{x,a}})) \right) \quad (3)$$

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

$$r_{u,x} = \frac{\mathcal{D}(\mathcal{E}(\sum_{a|a \neq x} \Delta_{x,a}) \prod_{a|a \neq x} (\mathcal{E}(r_{u,a})^{\phi_{x,a}}))}{\sum_{a|a \neq x} \phi_{x,a}} \quad (4)$$

Privacy preserving Slope One CF

- Since Δ and ϕ 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 CF

- Since Δ and ϕ 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 CF

- Since Δ and ϕ 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.**

Google App Engine implementation

Google App Engine for Java (GAE/J) features:

- **Automatically allocated scalable resources for growing user requests.**
- Slow high replication datastore access but fast distributed in-memory cache.
- Low CPU performance per application instance: affects cryptographic operations.

Google App Engine implementation

Google App Engine for Java (GAE/J) features:

- Automatically allocated scalable resources for growing user requests.
- **Slow high replication datastore access but fast distributed in-memory cache.**
- Low CPU performance per application instance: affects cryptographic operations.

Google App Engine implementation

Google App Engine for Java (GAE/J) features:

- Automatically allocated scalable resources for growing user requests.
- Slow high replication datastore access but fast distributed in-memory cache.
- **Low CPU performance per application instance: affects cryptographic operations.**

We measured some limitations of the GAE/J

A. Basu, J. Vaidya, T. Dimitrakos, H. Kikuchi, *Feasibility of a privacy preserving collaborative filtering scheme on the Google App Engine - a performance case study*, Proceedings of the 27th ACM Symposium on Applied Computing (SAC) Cloud Computing track, Trento, Italy, 2012.

Google App Engine implementation

Google App Engine for Java (GAE/J) features:

- Automatically allocated scalable resources for growing user requests.
- Slow high replication datastore access but fast distributed in-memory cache.
- Low CPU performance per application instance: affects cryptographic operations.

We measured some limitations of the GAE/J

A. Basu, J. Vaidya, T. Dimitrakos, H. Kikuchi, *Feasibility of a privacy preserving collaborative filtering scheme on the Google App Engine - a performance case study*, Proceedings of the 27th ACM Symposium on Applied Computing (SAC) Cloud Computing track, Trento, Italy, 2012.

Google App Engine implementation

Table: Comparison of typical prediction timings with the Google App Engine (on a good day!)

Bit size ^a	Vector size ^b	Prediction time ^c
1024	5	410ms
1024	10	825ms
2048	5	1900ms
2048	10	3500ms

^aPaillier cryptosystem modulus bit size, i.e. $|n|$.

^bSize of the encrypted rating vector.

^cOur experiments with the Amazon Elastic Beanstalk show that EBS is substantially faster.

Demo

- **Google App Engine implementation:**
<http://gaejppcf.appspot.com/>.
- Amazon Elastic Beanstalk implementation:
<http://gaejppcf.elasticbeanstalk.com/>.
- Simulated attack 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. But this fails to conclusively link ratings to users even when using a simple IPv4 NAT.

Demo

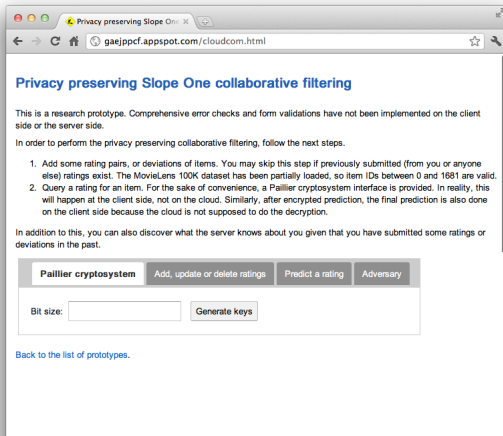
- Google App Engine implementation:
<http://gaejppcf.appspot.com/>.
- Amazon Elastic Beanstalk implementation:
<http://gaejppcf.elasticbeanstalk.com/>.
- Simulated attack on private data: in both cases, the cloud application tracks user's IPv4 address – a typical attack scenario to attempt to link ratings to users. But this fails to conclusively link ratings to users even when using a simple IPv4 NAT.

Demo

- Google App Engine implementation:
<http://gaejppcf.appspot.com/>.
- Amazon Elastic Beanstalk implementation:
<http://gaejppcf.elasticbeanstalk.com/>.
- **Simulated attack 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. But this fails to conclusively link ratings to users even when using a simple IPv4 NAT.**

Demo

Paillier cryptosystem helper:



The screenshot shows a web browser window with the address bar displaying "gaejppcf.appspot.com/cloudcom.html". The page title is "Privacy preserving Slope One collaborative filtering". The main content includes a disclaimer, instructions for using the tool, and a set of interactive buttons.

Privacy preserving Slope One collaborative filtering

This is a research prototype. Comprehensive error checks and form validations have not been implemented on the client side or the server side.

In order to perform the privacy preserving collaborative filtering, follow the next steps.

1. Add some rating pairs, or deviations of items. You may skip this step if previously submitted (from you or anyone else) ratings exist. The MovieLens 100K dataset has been partially loaded, so item IDs between 0 and 1681 are valid.
2. Query a rating for an item. For the sake of convenience, a Paillier cryptosystem interface is provided. In reality, this will happen at the client side, not on the cloud. Similarly, after encrypted prediction, the final prediction is also done on the client side because the cloud is not supposed to do the decryption.

In addition to this, you can also discover what the server knows about you given that you have submitted some ratings or deviations in the past.

Paillier cryptosystem Add, update or delete ratings Predict a rating Adversary

Bit size:

[Back to the list of prototypes.](#)

Demo

Add, update or delete ratings:

The screenshot shows a web browser window with the URL `gaejppcf.appspot.com/cloudcom.html`. The page title is "Privacy preserving Slope One collaborative filtering".

This is a research prototype. Comprehensive error checks and form validations have not been implemented on the client side or the server side.

In order to perform the privacy preserving collaborative filtering, follow the next steps.

1. Add some rating pairs, or deviations of items. You may skip this step if previously submitted (from you or anyone else) ratings exist. The MovieLens 100K dataset has been partially loaded, so item IDs between 0 and 1681 are valid.
2. Query a rating for an item. For the sake of convenience, a Paillier cryptosystem interface is provided. In reality, this will happen at the client side, not on the cloud. Similarly, after encrypted prediction, the final prediction is also done on the client side because the cloud is not supposed to do the decryption.

In addition to this, you can also discover what the server knows about you given that you have submitted some ratings or deviations in the past.

The interface features a navigation bar with four tabs: "Paillier cryptosystem", "Add, update or delete ratings" (which is active), "Predict a rating", and "Adversary".

Below the navigation bar, there is a checkbox labeled "Use experimental RPC self thread proxy to make your identity indeterminable." which is currently unchecked.

The "Item names:" field contains two empty text input boxes.

Below the item names, there are two radio buttons: "Ratings" (which is selected) and "Deviation".

The "Deviation diff:" field contains one empty text input box.

At the bottom right of the form, there are two buttons: "Update" and "Delete".

At the bottom left of the page, there is a link: [Back to the list of prototypes.](#)

Demo

Prediction query:

Privacy preserving Slope One collaborative filtering

This is a research prototype. Comprehensive error checks and form validations have not been implemented on the client side or the server side.

In order to perform the privacy preserving collaborative filtering, follow the next steps.

1. Add some rating pairs, or deviations of items. You may skip this step if previously submitted (from you or anyone else) ratings exist. The MovieLens 100K dataset has been partially loaded, so item IDs between 0 and 1681 are valid.
2. Query a rating for an item. For the sake of convenience, a Paillier cryptosystem interface is provided. In reality, this will happen at the client side, not on the cloud. Similarly, after encrypted prediction, the final prediction is also done on the client side because the cloud is not supposed to do the decryption.

In addition to this, you can also discover what the server knows about you given that you have submitted some ratings or deviations in the past.

Paillier cryptosystem Add, update or delete ratings **Predict a rating** Adversary

Item to query:

Given ratings:

Item and encrypted rating:

[Back to the list of prototypes.](#)

Demo

Adversary:

Privacy preserving Slope One collaborative filtering

This is a research prototype. Comprehensive error checks and form validations have not been implemented on the client side or the server side.

In order to perform the privacy preserving collaborative filtering, follow the next steps.

1. Add some rating pairs, or deviations of items. You may skip this step if previously submitted (from you or anyone else) ratings exist. The MovieLens 100K dataset has been partially loaded, so item IDs between 0 and 1681 are valid.
2. Query a rating for an item. For the sake of convenience, a Paillier cryptosystem interface is provided. In reality, this will happen at the client side, not on the cloud. Similarly, after encrypted prediction, the final prediction is also done on the client side because the cloud is not supposed to do the decryption.

In addition to this, you can also discover what the server knows about you given that you have submitted some ratings or deviations in the past.

Paillier cryptosystem Add, update or delete ratings Predict a rating **Adversary**

Reveal server audits

[Back to the list of prototypes.](#)

Conclusions

Our proposed scheme:

- is fast, accurate and easy to implement;
- uses user encrypted predicted query;
- does not store users' rating data;
- makes rating-to-user linkability hard through the use of anonymising techniques; and

Conclusions

Our proposed scheme:

- is fast, accurate and easy to implement;
- **uses user encrypted predicted query;**
- does not store users' rating data;
- makes rating-to-user linkability hard through the use of anonymising techniques; and
- scales well on real world PaaS clouds.

Conclusions

Our proposed scheme:

- is fast, accurate and easy to implement;
- uses user encrypted predicted query;
- **does not store users' rating data;**
- makes rating-to-user linkability hard through the use of anonymising techniques; and
- scales well on real world PaaS clouds.

Conclusions

Our proposed scheme:

- is fast, accurate and easy to implement;
- uses user encrypted predicted query;
- does not store users' rating data;
- **makes rating-to-user linkability hard through the use of anonymising techniques; and**
- scales well on real world PaaS clouds.

Conclusions

Our proposed scheme:

- is fast, accurate and easy to implement;
- uses user encrypted predicted query;
- does not store users' rating data;
- makes rating-to-user linkability hard through the use of anonymising techniques; and
- **scales well on real world PaaS clouds.**

Future work

- **Implement the proposal on vertical partition from our extended paper in the IEEE Cloudcom 2011.**
- Implement parallelism in prediction queries with large query vectors.
- Conduct comparative performance analyses with other privacy preserving CF implementations on different PaaS clouds.
- 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 from our extended paper in the IEEE Cloudcom 2011.
- **Implement parallelism in prediction queries with large query vectors.**
- Conduct comparative performance analyses with other privacy preserving CF implementations on different PaaS clouds.
- 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 from our extended paper in the IEEE Cloudcom 2011.
- Implement parallelism in prediction queries with large query vectors.
- **Conduct comparative performance analyses with other privacy preserving CF implementations on different PaaS clouds.**
- 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 from our extended paper in the IEEE Cloudcom 2011.
- Implement parallelism in prediction queries with large query vectors.
- Conduct comparative performance analyses with other privacy preserving CF implementations on different PaaS clouds.
- **Improve our scheme by discarding some assumptions (e.g. honest user) and dependencies (e.g. anonymiser networks).**

Thank you for listening!

Any questions?