# Privacy-preserving Slope One

Anirban Basu

Department of Electrical Engineering
Faculty of Engineering
Tokai University

TP-DIS (workshop at the IFIPTM 2011)
28 June 2011

# Table of Contents

# What are we doing?

We often have user-item rating data like this:

| | Virgin Atlantic | SAS | Singapore Airlines |
|---|---|---|---|
| Anirban | 3 | ? | 5 |
| Ian | 3 | 2 | 5 |
| Steve | 4 | 2 | 4 |

- The objective is to find the missing rating.
- A well-known technique is *collaborative filtering* (CF), which can be memory-based or model-based.

# What are we doing?

We often have user-item rating data like this:

| | Virgin Atlantic | SAS | Singapore Airlines |
|---|---|---|---|
| Anirban | 3 | ? | 5 |
| Ian | 3 | 2 | 5 |
| Steve | 4 | 2 | 4 |

- The objective is to find the missing rating.
- A well-known technique is *collaborative filtering* (CF), which can be memory-based or model-based.

# What are we doing?

Classification of collaborative filtering:

- Memory-based CF uses proximity (or distance) measures between ratings by users (i.e. *user-based*) or between ratings of items (i.e. *item-based*).
- Examples use, as similarity measures, cosine similarity, Pearson's Product-Moment Correlation Coefficient.
- Model-based CF *trains* a compact model from the user-item dataset and uses the model to predict.
- Examples use Singular Value Decomposition, Latent Semantic Indexing.

# What are we doing?

Classification of collaborative filtering:

- Memory-based CF uses proximity (or distance) measures between ratings by users (i.e. *user-based*) or between ratings of items (i.e. *item-based*).
- Examples use, as similarity measures, cosine similarity, Pearson's Product-Moment Correlation Coefficient.
- Model-based CF *trains* a compact model from the user-item dataset and uses the model to predict.
- Examples use Singular Value Decomposition, Latent Semantic Indexing.

# What are we doing?

Classification of collaborative filtering:

- Memory-based CF uses proximity (or distance) measures between ratings by users (i.e. *user-based*) or between ratings of items (i.e. *item-based*).
- Examples use, as similarity measures, cosine similarity, Pearson's Product-Moment Correlation Coefficient.
- Model-based CF *trains* a compact model from the user-item dataset and uses the model to predict.
- Examples use Singular Value Decomposition, Latent Semantic Indexing.

# What are we doing?

Classification of collaborative filtering:

- Memory-based CF uses proximity (or distance) measures between ratings by users (i.e. *user-based*) or between ratings of items (i.e. *item-based*).
- Examples use, as similarity measures, cosine similarity, Pearson's Product-Moment Correlation Coefficient.
- Model-based CF *trains* a compact model from the user-item dataset and uses the model to predict.
- Examples use Singular Value Decomposition, Latent Semantic Indexing.

# What are we doing?

So, what is the problem?

- Privacy: in our case, to compute a rating prediction through CF using multiple sites without leaking privacy of the data between the sites.
- Accuracy: do we trade accuracy for privacy?
- Efficiency: do we trade efficiency for privacy?

[1] I will explain this in a minute!

# What are we doing?

So, what is the problem?

- Privacy: in our case, to compute a rating prediction through CF using multiple sites without leaking privacy of the data between the sites.
- Accuracy: do we trade accuracy for privacy?
- Efficiency: do we trade efficiency for privacy?

[1] I will explain this in a minute!

# What are we doing?

So, what is the problem?

- Privacy: in our case, to compute a rating prediction through CF using multiple sites without leaking privacy of the data between the sites.
- Accuracy: do we trade accuracy for privacy?
- Efficiency: do we trade efficiency for privacy?

[1] I will explain this in a minute!

# What are we doing?

Our contribution

- First work to propose privacy preserving Slope One CF[1] for both horizontal and vertical data partitions.
- Proposed scheme is accurate with no leakage of private data; and
- at relatively low computational and communication complexity, i.e. it is achievable (implementation results say so).

[1] I will explain this in a minute!

# What are we doing?

Our contribution

- First work to propose privacy preserving Slope One CF[1] for both horizontal and vertical data partitions.
- Proposed scheme is accurate with no leakage of private data; and
- at relatively low computational and communication complexity, i.e. it is achievable (implementation results say so).

[1] I will explain this in a minute!

# What are we doing?

Our contribution

- First work to propose privacy preserving Slope One CF[1] for both horizontal and vertical data partitions.
- Proposed scheme is accurate with no leakage of private data; and
- at relatively low computational and communication complexity, i.e. it is achievable (implementation results say so).

[1] I will explain this in a minute!

# What is Slope One?

## Based on:

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 average deviations between ratings of items, weighted by relative cardinalities of pairs of items.
- Efficient: compared with user-based CF using cosine similarity and with singular value decomposition, Slope One scored the lowest mean absolute error (MAE) and root mean squared error (RMSE) on Apache Mahout reference

# 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 average deviations between ratings of items, weighted by relative cardinalities of pairs of items.
- Efficient: compared with user-based CF using cosine similarity and with singular value decomposition, Slope One scored the lowest mean absolute error (MAE) and root mean squared error (RMSE) on Apache Mahout reference implementation for MovieLens 100K and 1M datasets.

# 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 average deviations between ratings of items, weighted by relative cardinalities of pairs of items.
- Efficient: compared with user-based CF using cosine similarity and with singular value decomposition, Slope One scored the lowest mean absolute error (MAE) and root mean squared error (RMSE) on Apache Mahout reference implementation for MovieLens 100K and 1M datasets.

# 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 average deviations between ratings of items, weighted by relative cardinalities of pairs of items.
- Efficient: compared with user-based CF using cosine similarity and with singular value decomposition, Slope One scored the lowest mean absolute error (MAE) and root mean squared error (RMSE) on Apache Mahout reference implementation for MovieLens 100K and 1M datasets.

# CF algorithm efficiency (MovieLens 100K ratings)

CF algorithms reference implementations using about 70% users and 90% preferences:

- **Random**[2]: $MAE = 1.3572$, $RMSE = 1.6731$.
- **User similarity based (cosine similarity)**: $MAE = 0.8461$, $RMSE = 1.09685$.
- **Singular Value Decomposition (SVD)**: $MAE = 0.7578$, $RMSE = 0.9523$.
- **Slope One**: $MAE = 0.7350$, $RMSE = 0.9268$.

[2] Only used for benchmarking

# CF efficiency (MovieLens 1M ratings)

CF algorithms reference implementations using about 30% users and 90% preferences[3]:

- **Random**: $MAE = 1.3954$, $RMSE = 1.7216$.
- **User similarity based (cosine similarity)**: $MAE = 0.8399$, $RMSE = 1.0858$.
- **Singular Value Decomposition (SVD)**: $MAE = 0.7167$, $RMSE = 0.9180$.
- **Slope One**: $MAE = 0.7135$, $RMSE = 0.9019$.

[3] Similar to the 100K dataset, the MAE and RMSE values can vary depending on parameters used for the various CF algorithms.

# Rationale behind Slope One

Let's rate the airlines companies – British Airways and Emirates:

| | British Airways | Emirates |
|---|---|---|
| BASU Anirban | 1 | ? |
| KIKUCHI Hiroaki | 2 | 4 |
| VAIDYA Jaideep | 2 | 5 |

$$? = ((4 - 2) + (5 - 2))/2 + 1 = 3.5$$

- So, Anirban's rating for Emirates is his rating for British Airways added to the average difference of ratings from Emirates to British Airways for users where ratings are present.

# Rationale behind Slope One

Let's rate the airlines companies – British Airways and Emirates:

| | British Airways | Emirates |
|---|---|---|
| BASU Anirban | 1 | ? |
| KIKUCHI Hiroaki | 2 | 4 |
| VAIDYA Jaideep | 2 | 5 |

$$? = ((4 - 2) + (5 - 2))/2 + 1 = 3.5$$

- So, Anirban's rating for Emirates is his rating for British Airways added to the average difference of ratings from Emirates to British Airways for users where ratings are present.

# Rationale behind Slope One

When we add another airline (Cathay Pacific), we get:

| | British Airways | Emirates | Cathay Pacific |
|---|---|---|---|
| BASU Anirban | 1 | ? | 4 |
| KIKUCHI Hiroaki | 2 | 4 | 4 |
| VAIDYA Jaideep | 2 | 5 | 4 |

$$? = \frac{(\frac{(4-2)+(5-2)}{2} + 1) + (\frac{(4-4)+(5-4)}{2} + 4)}{2} = 4.0$$

- If we reason item-based predictions in this way then we are performing *unweighted* Slope One.

# Rationale behind Slope One

When we add another airline (Cathay Pacific), we get:

| | British Airways | Emirates | Cathay Pacific |
|---|---|---|---|
| BASU Anirban | 1 | ? | 4 |
| KIKUCHI Hiroaki | 2 | 4 | 4 |
| VAIDYA Jaideep | 2 | 5 | 4 |

$$? = \frac{(\frac{(4-2)+(5-2)}{2} + 1) + (\frac{(4-4)+(5-4)}{2} + 4)}{2} = 4.0$$

- If we reason item-based predictions in this way then we are performing *unweighted* Slope One.

# 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:

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

# Pre-computation

Therefore, Slope One predictors need two things pre-computed before accurate predictions can be made.

- ***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-computation

Therefore, Slope One predictors need two things pre-computed before accurate predictions can be made.

- ***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 $\Delta$.

# Preserving privacy with Slope One CF

- The scheme we propose involves an *additively homomorphic public key cryptosystem*, e.g. Paillier, Damgärd-Jurik, i.e. homomorphic addition (we denote encryption and decryption functions as $\mathcal{E}()$ and $\mathcal{D}()$ respectively):

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

and homomorphic multiplication:

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

- Since using encryption, we do not trade accuracy for privacy; and we have efficiency advantages due to the pre-computation stage.

# Preserving privacy with Slope One CF

- The scheme we propose involves an *additively homomorphic public key cryptosystem*, e.g. Paillier, Damgärd-Jurik, i.e. homomorphic addition (we denote encryption and decryption functions as $\mathcal{E}()$ and $\mathcal{D}()$ respectively):

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

  and homomorphic multiplication:

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

- Since using encryption, we do not trade accuracy for privacy; and we have efficiency advantages due to the pre-computation stage.

# Preserving privacy with Slope One CF

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

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

and thus, the final prediction is given as:

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

# Preserving privacy with Slope One CF

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

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

and thus, the final prediction is given as:

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

# Preserving privacy with Slope One CF

- Proposed solution supports encrypted prediction query and response; and encryption of the deviation matrix $\Delta$.
- Although $\Delta$ and $\phi$ are not private information with respect to user data, an attacker with knowledge of unencrypted values of both can run brute-force queries to guess the users' actual ratings.

# Preserving privacy with Slope One CF

- Proposed solution supports encrypted prediction query and response; and encryption of the deviation matrix $\Delta$.
- Although $\Delta$ and $\phi$ are not private information with respect to user data, an attacker with knowledge of unencrypted values of both can run brute-force queries to guess the users' actual ratings.

# Preserving privacy with Slope One CF

- The $\mathcal{E}(\Delta)$ and $\phi$ matrices are pre-computed.
- An encrypted query is run and the result is decrypted using threshold decryption keys.
- Note that the encrypted query can be answered by any of the collaborating sites without leakage of private information.

# Preserving privacy with Slope One CF

- The $\mathcal{E}(\Delta)$ and $\phi$ matrices are pre-computed.
- An encrypted query is run and the result is decrypted using threshold decryption keys.
- Note that the encrypted query can be answered by any of the collaborating sites without leakage of private information.

# Preserving privacy with Slope One CF

- The $\mathcal{E}(\Delta)$ and $\phi$ matrices are pre-computed.
- An encrypted query is run and the result is decrypted using threshold decryption keys.
- Note that the encrypted query can be answered by any of the collaborating sites without leakage of private information.

# Pictorial representations of prediction and pre-computation

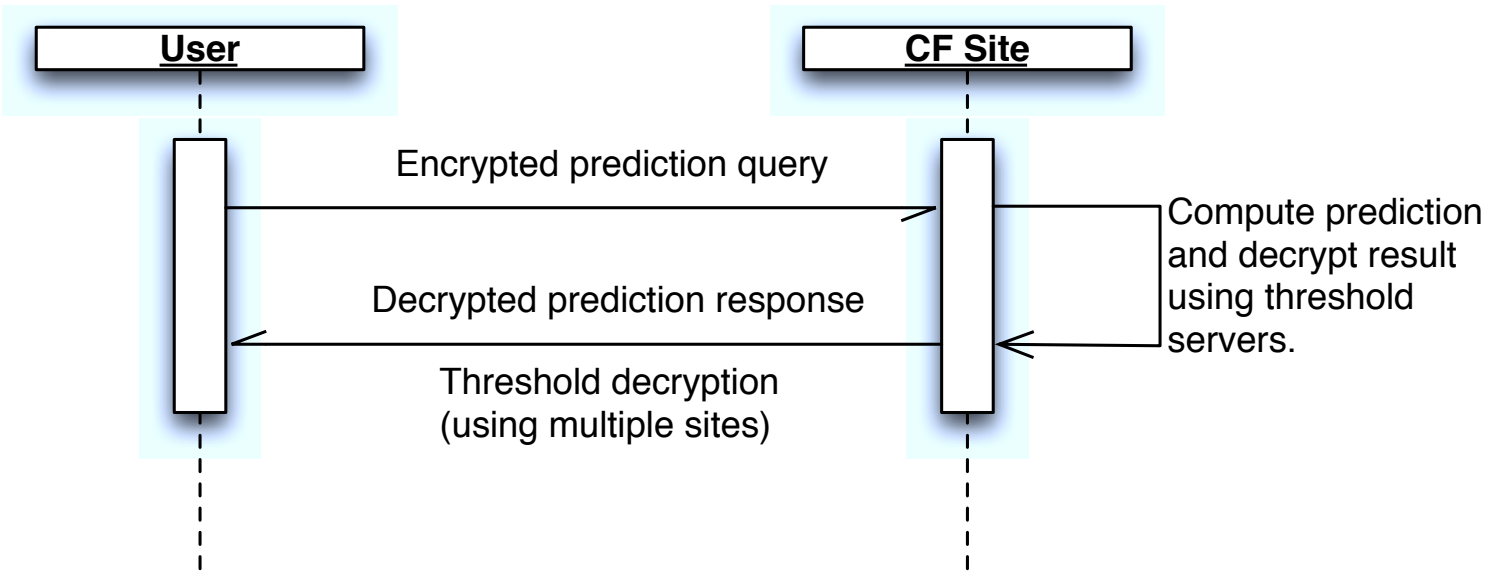


Figure: **UML sequence diagram for prediction of between any one user and a site.**

# Pictorial representations of prediction and pre-computation

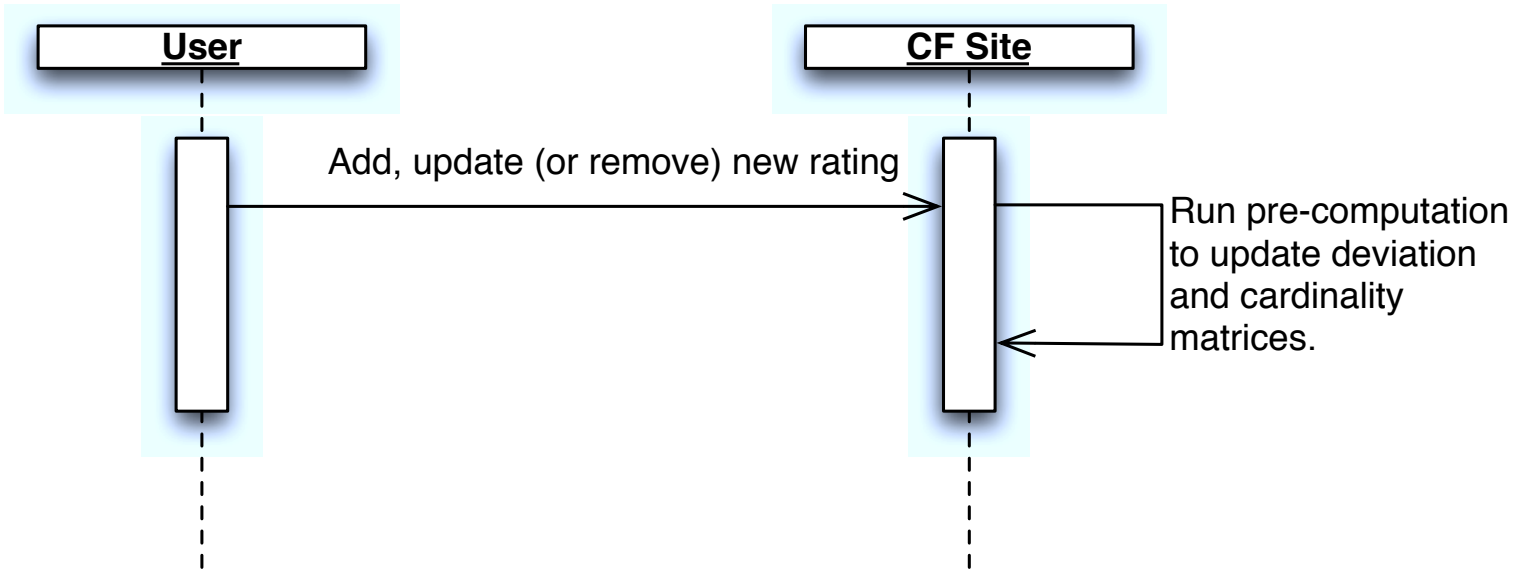


Figure: **UML sequence diagram for addition, update or deletion of data between any one user and a site.**

# Horizontal and vertical partitioning

Horizontal partition:

- Pure horizontal partition: completely disjoint user-set but fully joined item-set.
- Aggregation from multiple sites: add (or homomorphically add) the deviation and cardinality matrices from the different sites.
- Pre-computation requires $O(kn^2)$ encryptions for $k$ sites and $n$ items. $O(w)$ prediction complexity as the prediction is dominated by the decryption operations at $w$ threshold servers.

# Horizontal and vertical partitioning

Horizontal partition:

- Pure horizontal partition: completely disjoint user-set but fully joined item-set.
- Aggregation from multiple sites: add (or homomorphically add) the deviation and cardinality matrices from the different sites.
- Pre-computation requires $O(kn^2)$ encryptions for $k$ sites and $n$ items. $O(w)$ prediction complexity as the prediction is dominated by the decryption operations at $w$ threshold servers.

# Horizontal and vertical partitioning

Horizontal partition:

- Pure horizontal partition: completely disjoint user-set but fully joined item-set.
- Aggregation from multiple sites: add (or homomorphically add) the deviation and cardinality matrices from the different sites.
- Pre-computation requires $O(kn^2)$ encryptions for $k$ sites and $n$ items. $O(w)$ prediction complexity as the prediction is dominated by the decryption operations at $w$ threshold servers.

# Horizontal and vertical partitioning

Vertical partition:

- Pure vertical partition: completely disjoint item-set but fully joined user-set.
- Non-trivial aggregation using the secure scalar product protocol (Vaidya and Clifton). (And, I won't bore you with the details, which are in the paper!)
- Pre-computation requires $O(mn^2)$ encryptions for $m$ users and $n$ items. Again, $O(w)$ prediction complexity.

# Horizontal and vertical partitioning

Vertical partition:

- Pure vertical partition: completely disjoint item-set but fully joined user-set.
- Non-trivial aggregation using the secure scalar product protocol (Vaidya and Clifton). (And, I won't bore you with the details, which are in the paper!)
- Pre-computation requires $O(mn^2)$ encryptions for $m$ users and $n$ items. Again, $O(w)$ prediction complexity.

# Horizontal and vertical partitioning

Vertical partition:

- Pure vertical partition: completely disjoint item-set but fully joined user-set.
- Non-trivial aggregation using the secure scalar product protocol (Vaidya and Clifton). (And, I won't bore you with the details, which are in the paper!)
- Pre-computation requires $O(mn^2)$ encryptions for $m$ users and $n$ items. Again, $O(w)$ prediction complexity.

# Pre-computation over generic partitions

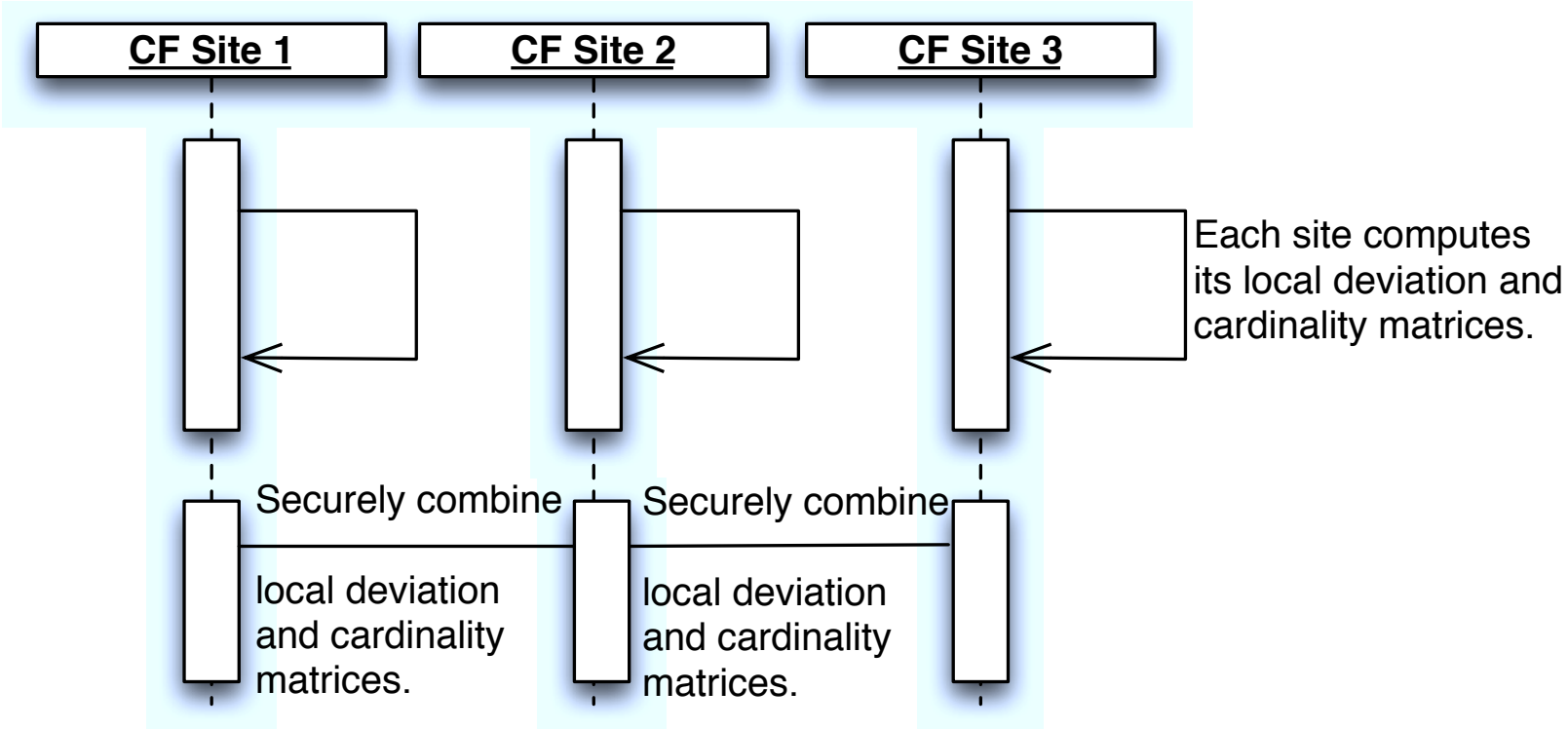


Figure: **UML sequence diagram describing aggregation over generic partitions during the pre-computation stage.**

# How does it compare?

| | CF type | Accuracy | Scalability |
|---|---|---|---|
| Shuguo Han, et. al. (2005) | Model-based[4] | High | Low |
| John Canny (2002) | Model-based[5] | High | High |
| Our scheme (2011) | Item-based | High | High[6] |

Table: Comparison of our scheme with some other CF schemes that use encryption to preserve privacy.

[4] Homomorphic encryption with SVD, QR decomposition.
[5] partial SVD and homomorphic encryption.
[6] Tested on dataset different from Canny's work.

# Single-site, single-partition implementation

**Something not present in the paper!**

Our paper does not have implementation results beyond estimated evaluation of time complexity of our algorithms. Here we present some initial implementation results.

# Single-site, single-partition implementation

- Multi-threaded, single-site, single-server initial implementation in Java 1.6.
- In-memory matrix storage in synchronized `HashMap < K1, HashMap< K2, V > >` using `Integer` keys for $O(1)$ lookup and storage.
- Main advantage of our CF scheme: low absolute-error prediction available during pre-computation.

**Fast prediction**

Fast prediction, during pre-computation (which gets faster after pre-computation): we show some results in the next slide.

# Single-site, single-partition implementation

- Multi-threaded, single-site, single-server initial implementation in Java 1.6.
- In-memory matrix storage in synchronized `HashMap < K1, HashMap< K2, V > >` using `Integer` keys for $O(1)$ lookup and storage.
- Main advantage of our CF scheme: low absolute-error prediction available during pre-computation.

**Fast prediction**

Fast prediction, during pre-computation (which gets faster after pre-computation): we show some results in the next slide.

# Single-site, single-partition implementation

- Multi-threaded, single-site, single-server initial implementation in Java 1.6.
- In-memory matrix storage in synchronized `HashMap < K1, HashMap< K2, V > >` using `Integer` keys for $O(1)$ lookup and storage.
- Main advantage of our CF scheme: low absolute-error prediction available during pre-computation.

**Fast prediction**

Fast prediction, during pre-computation (which gets faster after pre-computation): we show some results in the next slide.

# Single-site, single-partition implementation

- Multi-threaded, single-site, single-server initial implementation in Java 1.6.
- In-memory matrix storage in synchronized `HashMap < K1, HashMap< K2, V > >` using `Integer` keys for $O(1)$ lookup and storage.
- Main advantage of our CF scheme: low absolute-error prediction available during pre-computation.

**Fast prediction**

Fast prediction, during pre-computation (which gets faster after pre-computation): we show some results in the next slide.

# Single-site, single-partition implementation

Hardware 2.53GHz Intel Core 2 Duo (dual core) processor, 8GB RAM, Mac OS X 10.6.7 and 64-bit Java 1.6.

| **Keys**[a] | **Mean PC**[b] | **Total PC**[c] | **Prediction**[d] |
|---|---|---|---|
| 512 bits | 30s | 2h | 500ms |
| 1024 bits | 90s | 6h | 1.5s |
| 2048 bits | 270s | 18h | 4.5s |

[a] This refers to modulus bit length of the cryptosystem.

[b] This is the mean of the pre-computation time taken for each user and all item pairs.

[c] This is an approximate linear estimate. Note that the total is not equal to the mean time for pre-computation multiplied by the number of users because the pre-computation tasks run in parallel.

[d] Note that prediction times drop to about a tenth after pre-computation!

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:
  - is easy to implement;
  - is fast;
  - allows for prediction during pre-computation;

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:
  - is easy to implement;
  - is fast;
  - allows for prediction during pre-computation;
  - supports encrypted query to any collaborating site; and

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:
  - is easy to implement;
  - is fast;
  - allows for prediction during pre-computation;
  - supports encrypted query to any collaborating site; and
  - scales well.

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:
  - is easy to implement;
  - is fast;
  - allows for prediction during pre-computation;
  - supports encrypted query to any collaborating site; and
  - scales well.

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:
  - is easy to implement;
  - is fast;
  - allows for prediction during pre-computation;
  - supports encrypted query to any collaborating site; and
  - scales well.

# Conclusions

- Our work is the only one that uses encryption to preserve privacy on Slope One CF.
- The proposed scheme:
  - is easy to implement;
  - is fast;
  - allows for prediction during pre-computation;
  - supports encrypted query to any collaborating site; and
  - scales well.

# Future work

- Speeding up pre-computation with better parallelism (currently using thread level parallelism, which may not be ideal in distributed environments).
- Mixed data partitions.
- Can we reduce computational complexity of privacy preservation in favour of trust?

# Future work

- Speeding up pre-computation with better parallelism (currently using thread level parallelism, which may not be ideal in distributed environments).
- Mixed data partitions.
- Can we reduce computational complexity of privacy preservation in favour of trust?

# Future work

- Speeding up pre-computation with better parallelism (currently using thread level parallelism, which may not be ideal in distributed environments).
- Mixed data partitions.
- Can we reduce computational complexity of privacy preservation in favour of trust?

**Something interesting to look into:**

Seigneur, J. M., Jensen, C. D. 2004. *Trading Privacy for Trust*. In: Proc. iTrust 2004 (Springer).

# Future work

- Speeding up pre-computation with better parallelism (currently using thread level parallelism, which may not be ideal in distributed environments).
- Mixed data partitions.
- Can we reduce computational complexity of privacy preservation in favour of trust?

**Something interesting to look into:**

Seigneur, J. M., Jensen, C. D. 2004. *Trading Privacy for Trust*. In: Proc. iTrust 2004 (Springer).

# Thank you for listening!

Any questions?