Practical privacy using homomorphic encryption – a myth or reality?

Anirban Basu

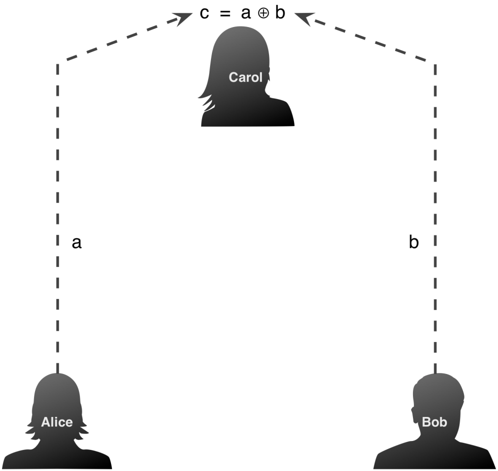
Department of Electrical Engineering
Faculty of Engineering
Tokai University (Japan)

SecureCloud 2012
May 9, 2012
Frankfurt am Main, Germany

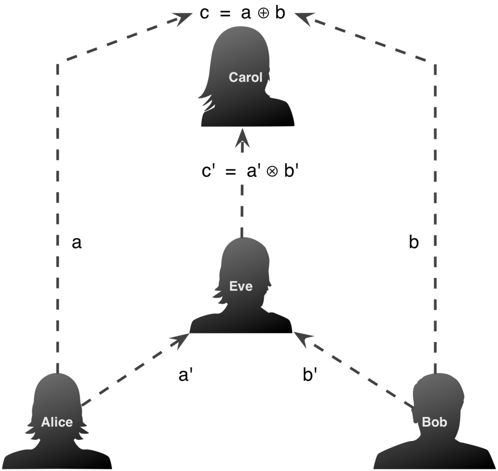
The magic of homomorphic encryption

1. Homomorphic encryption
   - What and why?
   - Application in recommender systems
2. Privacy preserving collaborative filtering (PPCF)
   - Collaborative filtering (CF), briefly
   - Privacy preserving Slope One
3. On the SaaS cloud
   - Google App Engine feasibility
   - An alternative approach
4. Tail piece
   - Conclusions and future avenues
   - Question time!

Compute blindly?



Compute blindly?



Compute blindly?

$$(m_1 \oplus m_2)' = m_1' \otimes m_2'$$

m_1 and m_2: plaintext messages; x': ciphertext of x; $\oplus$ and $\otimes$: two operations that satisfy this ***homomorphic*** relation.

Homomorphic cryptosystems

Depending on $\oplus$ and $\otimes$, we have:

- additive (e.g., Paillier, Damgärd-Jurik, Elliptic Curve ElGamal),
- multiplicative (e.g., RSA, ElGamal), and
- fully homomorphic cryptosystems (Craig Gentry et al.).

- too good to be true?

Homomorphic cryptosystems

Depending on $\oplus$ and $\otimes$, we have:

- additive (e.g., Paillier, Damgärd-Jurik, Elliptic Curve ElGamal),
- multiplicative (e.g., RSA, ElGamal), and
- fully homomorphic cryptosystems (Craig Gentry et al.).
- too good to be true?
- or, practically feasible?

Homomorphic cryptosystems

Depending on $\oplus$ and $\otimes$, we have:

- additive (e.g., Paillier, Damgärd-Jurik, Elliptic Curve ElGamal),
- multiplicative (e.g., RSA, ElGamal), and
- fully homomorphic cryptosystems (Craig Gentry et al.).

- too good to be true?
- or, practically feasible?

Homomorphic cryptosystems

Depending on $\oplus$ and $\otimes$, we have:

- additive (e.g., Paillier, Damgärd-Jurik, Elliptic Curve ElGamal),
- multiplicative (e.g., RSA, ElGamal), and
- fully homomorphic cryptosystems (Craig Gentry et al.).

but, is this *magic*

- too good to be true?
- or, practically feasible?

Homomorphic cryptosystems

Depending on $\oplus$ and $\otimes$, we have:

- additive (e.g., Paillier, Damgärd-Jurik, Elliptic Curve ElGamal),
- multiplicative (e.g., RSA, ElGamal), and
- fully homomorphic cryptosystems (Craig Gentry et al.).

but, is this *magic*

- too good to be true?
- or, practically feasible?

Recommendation through collaborative filtering

- *'People who have bought this have also bought these'* – recommendation.
- Collaborative filtering (CF) – recommendation from opinions of the community.
- What about privacy in rating based collaborative filtering?

Recommendation through collaborative filtering

- *'People who have bought this have also bought these'* – recommendation.
- Collaborative filtering (CF) – recommendation from opinions of the community.
- What about privacy in rating based collaborative filtering?

Recommendation through collaborative filtering

- *'People who have bought this have also bought these'* – recommendation.
- Collaborative filtering (CF) – recommendation from opinions of the community.
- What about privacy in rating based collaborative filtering?

Privacy and recommender systems

- Can *someone* (the cloud?) compute CF for users?
- . . . and do so blindly?
- Privacy preserving collaborative filtering (PPCF) for the Software-as-a-Service cloud.

Privacy and recommender systems

- Can *someone* (the cloud?) compute CF for users?
- …and do so blindly?
- Privacy preserving collaborative filtering (PPCF) for the Software-as-a-Service cloud.

Privacy and recommender systems

- Can *someone* (the cloud?) compute CF for users?
- . . . and do so blindly?
- Privacy preserving collaborative filtering (PPCF) for the Software-as-a-Service cloud.

Application scenario

1 Homomorphic encryption
- What and why?
- Application in recommender systems

2 Privacy preserving collaborative filtering (PPCF)
- Collaborative filtering (CF), briefly
- Privacy preserving Slope One

3 On the SaaS cloud
- Google App Engine feasibility
- An alternative approach

4 Tail piece
- Conclusions and future avenues
- Question time!

CF – a form of recommendation

User-item rating data like this[1]:

	Canon 7D	Leica M9	Nikon D7000	...	Olympus OM-D
Alice	5	4	-	...	3
Bob	3	5	2	...	-
Carol	-	?	4	...	3
Dave	4	3	-	...	-

- The objective is to find a rating for Leica M9 for Carol.
- CF – a well-known recommendation technique, based on the preferences of the community.

[1] "-" indicates the absence of a rating.

CF – a form of recommendation

User-item rating data like this[1]:

	Canon 7D	Leica M9	Nikon D7000	...	Olympus OM-D
Alice	5	4	-	...	3
Bob	3	5	2	...	-
Carol	-	?	4	...	3
Dave	4	3	-	...	-

- The objective is to find a rating for Leica M9 for Carol.
- CF – a well-known recommendation technique, based on the preferences of the community.

[1] "-" indicates the absence of a rating.

Slope One predictors

Based on:

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

Slope One predictors

- CF predictors of the form $f(x) = x + b$, hence "slope one".
- Simple and efficient (compared with SVD, PMCC, Cosine).

Slope One predictors

- CF predictors of the form $f(x) = x + b$, hence "slope one".
- Simple and efficient (compared with SVD, PMCC, Cosine).

Slope One predictors

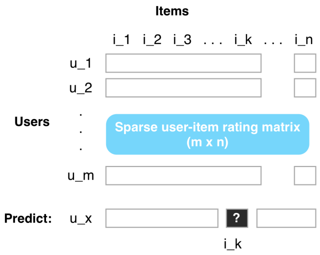


Figure: **The general CF problem.**

CF and Slope One

Pre-compute:

- ***Deviation matrix*** Δ: deviation of ratings of an item pair by the same user; dimension: $n \times n$.
- ***Cardinality matrix*** ϕ: number of co-existing ratings by the same user of an item pair; dimension same as Δ.

CF and Slope One

Pre-compute:

- ***Deviation matrix*** Δ: deviation of ratings of an item pair by the same user; dimension: $n \times n$.
- ***Cardinality matrix*** ϕ: number of co-existing ratings by the same user of an item pair; dimension same as Δ.

CF and Slope One

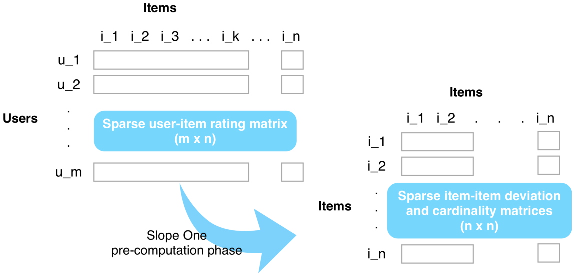


Figure: **Slope One pre-computation creates a 'model' which is used for prediction.**

The weighted Slope One predictor

- Average deviation:

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

$\phi_{a,b}$: the number of the users who have rated both items; $\delta_{i,a,b} = r_{i,a} - r_{i,b}$: the deviation of the ratings of item a from that of item b both given by user i.

- The *weighted* Slope One predictor:

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

The weighted Slope One predictor

- Average deviation:

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

$\phi_{a,b}$: the number of the users who have rated both items; $\delta_{i,a,b} = r_{i,a} - r_{i,b}$: the deviation of the ratings of item *a* from that of item *b* both given by user *i*.

- The *weighted* Slope One predictor:

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

Preserving privacy with Slope One CF

- An *additively homomorphic cryptosystem* – the Damgärd-Jurik cryptosystem, defining
  - homomorphic addition:

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

  - and homomorphic multiplication:

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

  m_1 and m_2 are plaintexts and π is a plaintext multiplicand.

Preserving privacy with Slope One CF

- An *additively homomorphic cryptosystem* – the Damgärd-Jurik cryptosystem, defining
  - homomorphic addition:

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

  - and homomorphic multiplication:

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

m_1 and m_2 are plaintexts and π is a plaintext multiplicand.

Preserving privacy with Slope One CF

- An *additively homomorphic cryptosystem* – the Damgärd-Jurik cryptosystem, defining
  - homomorphic addition:

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

  - and homomorphic multiplication:

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

m_1 and m_2 are plaintexts and π is a plaintext multiplicand.

Privacy preserving weighted Slope One predictors

- The numerator of the prediction equation:

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

- The final prediction:

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

Privacy preserving weighted Slope One predictors

- The numerator of the prediction equation:

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

- The final prediction:

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

Privacy preserving weighted Slope One predictors

- Pre-computed: $\mathcal{E}(\Delta)$ and ϕ.
- Encrypted query result decrypted using threshold decryption keys.
- Query answered by collaborating sites.
- Supports horizontal and vertical partitioning, using secure scalar product (Vaidya and Clifton).

Privacy preserving weighted Slope One predictors

- Pre-computed: $\mathcal{E}(\Delta)$ and ϕ.
- Encrypted query result decrypted using threshold decryption keys.
- Query answered by collaborating sites.
- Supports horizontal and vertical partitioning, using secure scalar product (Vaidya and Clifton).

Privacy preserving weighted Slope One predictors

- Pre-computed: $\mathcal{E}(\Delta)$ and ϕ.
- Encrypted query result decrypted using threshold decryption keys.
- Query answered by collaborating sites.
- Supports horizontal and vertical partitioning, using secure scalar product (Vaidya and Clifton).

Privacy preserving weighted Slope One predictors

- Pre-computed: $\mathcal{E}(\Delta)$ and ϕ.
- Encrypted query result decrypted using threshold decryption keys.
- Query answered by collaborating sites.
- Supports horizontal and vertical partitioning, using secure scalar product (Vaidya and Clifton).

Some results

- **Hardware:** 2.53GHz Intel Core 2 Duo (dual core) processor, 8GB RAM, Mac OS X 10.6.7 and 64-bit Java 1.6.
- **Implementation:** Single server, single partition, MovieLens 100K dataset.

Some results

- **Hardware:** 2.53GHz Intel Core 2 Duo (dual core) processor, 8GB RAM, Mac OS X 10.6.7 and 64-bit Java 1.6.
- **Implementation:** Single server, single partition, MovieLens 100K dataset.

Key size	Mean PC[a]	Total PC	Prediction
512 bits	30s	2h	500ms
1024 bits	90s	6h	1.5s
2048 bits	270s	18h	4.5s

[a] PC: pre-computation

Further reading

JISIS 2011 paper:

Basu, A., Vaidya, J., Kikuchi, H. 2011. *Efficient privacy-preserving collaborative filtering based on the weighted Slope One predictor*. In: Journal of Internet Services and Information Security (special edition).

Try this on the cloud

1. Homomorphic encryption
   - What and why?
   - Application in recommender systems
2. Privacy preserving collaborative filtering (PPCF)
   - Collaborative filtering (CF), briefly
   - Privacy preserving Slope One
3. On the SaaS cloud
   - Google App Engine feasibility
   - An alternative approach
4. Tail piece
   - Conclusions and future avenues
   - Question time!

Google App Engine for Java (GAE/J)

- A Software-as-a-Service (SaaS) engine.
- On-demand transparent scalability with low costs, including a daily free quota.
- Java servlet based computation model; allows for batch computations using task queues and cron.

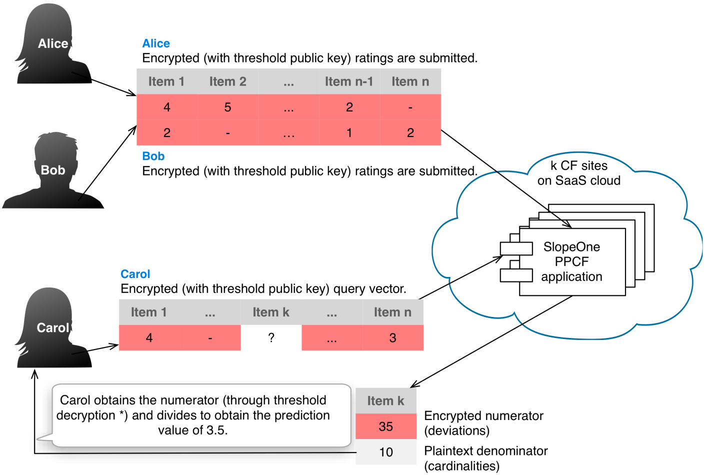
Google App Engine for Java (GAE/J)

- A Software-as-a-Service (SaaS) engine.
- On-demand transparent scalability with low costs, including a daily free quota.
- Java servlet based computation model; allows for batch computations using task queues and cron.

Google App Engine for Java (GAE/J)

- A Software-as-a-Service (SaaS) engine.
- On-demand transparent scalability with low costs, including a daily free quota.
- Java servlet based computation model; allows for batch computations using task queues and cron.

PPCF on the GAE/J



* Multiple PPCF sites compute the threshold decryption.

GAE/J challenges

- Servlet execution time limit (30s).
- High replication, slow access datastore.
- Lack of concurrency support (no threads!).
- Lack of control over resource allocation (bit better now with front-end instance classes).

GAE/J challenges

- Servlet execution time limit (30s).
- High replication, slow access datastore.
- Lack of concurrency support (no threads!).
- Lack of control over resource allocation (bit better now with front-end instance classes).

GAE/J challenges

- Servlet execution time limit (30s).
- High replication, slow access datastore.
- Lack of concurrency support (no threads!).
- Lack of control over resource allocation (bit better now with front-end instance classes).

GAE/J challenges

- Servlet execution time limit (30s).
- High replication, slow access datastore.
- Lack of concurrency support (no threads!).
- Lack of control over resource allocation (bit better now with front-end instance classes).

Further reading

ACM SAC 2012 paper:

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

An alternative PPCF on the GAE/J

- Identity anonymizer, plaintext deviation aggregation.
- No threshold decryption – user encrypts and decrypts.
- Prediction:

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

An alternative PPCF on the GAE/J

- Identity anonymizer, plaintext deviation aggregation.
- No threshold decryption – user encrypts and decrypts.
- Prediction:

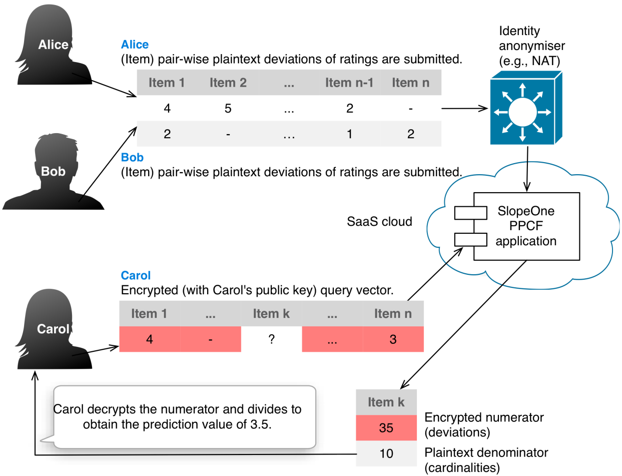
$$r_{u,x} = \frac{\mathcal{D}(\mathcal{E}(\sum_{a|a\neq x}\Delta_{x,a})\prod_{a|a\neq x}(\mathcal{E}(r_{u,a})^{\phi_{x,a}}))}{\sum_{a|a\neq x}\phi_{x,a}}$$

An alternative PPCF on the GAE/J

- Identity anonymizer, plaintext deviation aggregation.
- No threshold decryption – user encrypts and decrypts.
- Prediction:

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

An alternative PPCF on the GAE/J

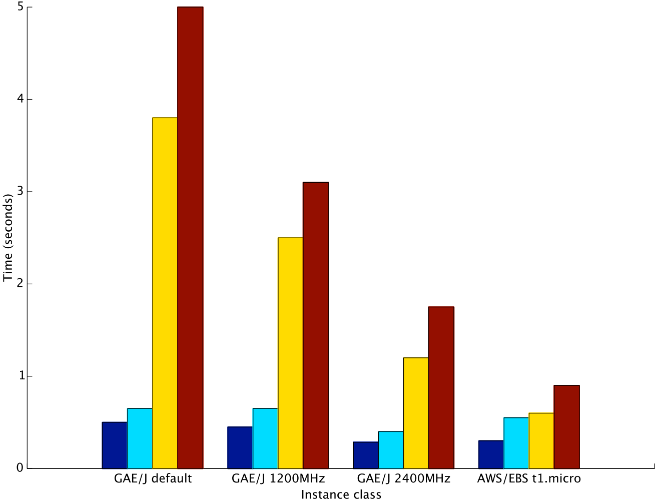


Results on the Google App Engine (GAE/J)

Bit size[a]	**Vector size**[b]	**Prediction time**
1024	5	500ms
1024	10	650ms
2048	5	3800ms
2048	10	5000ms

[a] Paillier cryptosystem modulus bit size, i.e. $|n|$.
[b] Size of the encrypted rating query vector.

Results: GAE/J and Amazon Elastic Beanstalk



Further reading

IEEE Cloudcom 2011 paper:

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

Let's wrap up

1 Homomorphic encryption
- What and why?
- Application in recommender systems

2 Privacy preserving collaborative filtering (PPCF)
- Collaborative filtering (CF), briefly
- Privacy preserving Slope One

3 On the SaaS cloud
- Google App Engine feasibility
- An alternative approach

4 Tail piece
- Conclusions and future avenues
- Question time!

Conclusions

Privacy with homomorphic encryption – worth it?

It works well in certain scenarios albeit some, mainly, performance issues.

Conclusions

- Homomorphic encryption challenges: threshold decryption, computational complexity, supported operations.
- Application areas: privacy preserving data mining, graph problems e.g., resource allocation.
- Cloud applicability: performance, feasibility depends on type of application, platform.

Conclusions

- Homomorphic encryption challenges: threshold decryption, computational complexity, supported operations.
- Application areas: privacy preserving data mining, graph problems e.g., resource allocation.
- Cloud applicability: performance, feasibility depends on type of application, platform.

Conclusions

- Homomorphic encryption challenges: threshold decryption, computational complexity, supported operations.
- Application areas: privacy preserving data mining, graph problems e.g., resource allocation.
- Cloud applicability: performance, feasibility depends on type of application, platform.

Privacy, cloud, responsibility and the future

- Better performance – dedicated cryptographic hardware?
- Optimisation for computational complexity, power use – cloud infrastructure consumes power.
- Alternatives – open question?

Privacy, cloud, responsibility and the future

- Better performance – dedicated cryptographic hardware?
- Optimisation for computational complexity, power use – cloud infrastructure consumes power.
- Alternatives – open question?

Privacy, cloud, responsibility and the future

- Better performance – dedicated cryptographic hardware?
- Optimisation for computational complexity, power use – cloud infrastructure consumes power.
- Alternatives – open question?

Thank you for listening!

Any questions?