# Private Decentralized Recommendation

Davide Frey davide.frey@inria.fr
WIDE Team

INRIA Rennes

# Outline

- Decentralized Recommendation
- Privacy by Profile Blurring
- Privacy by Proxy
- Privacy through Landmarks

# Clustering similar peers

- Vicinity: Introducing application-dependent proximity metric [VvS, EuroPar 2005]
- Two-layered approach
  - Biased gossip reflecting some application semantic
  - Unbiased peer sampling service

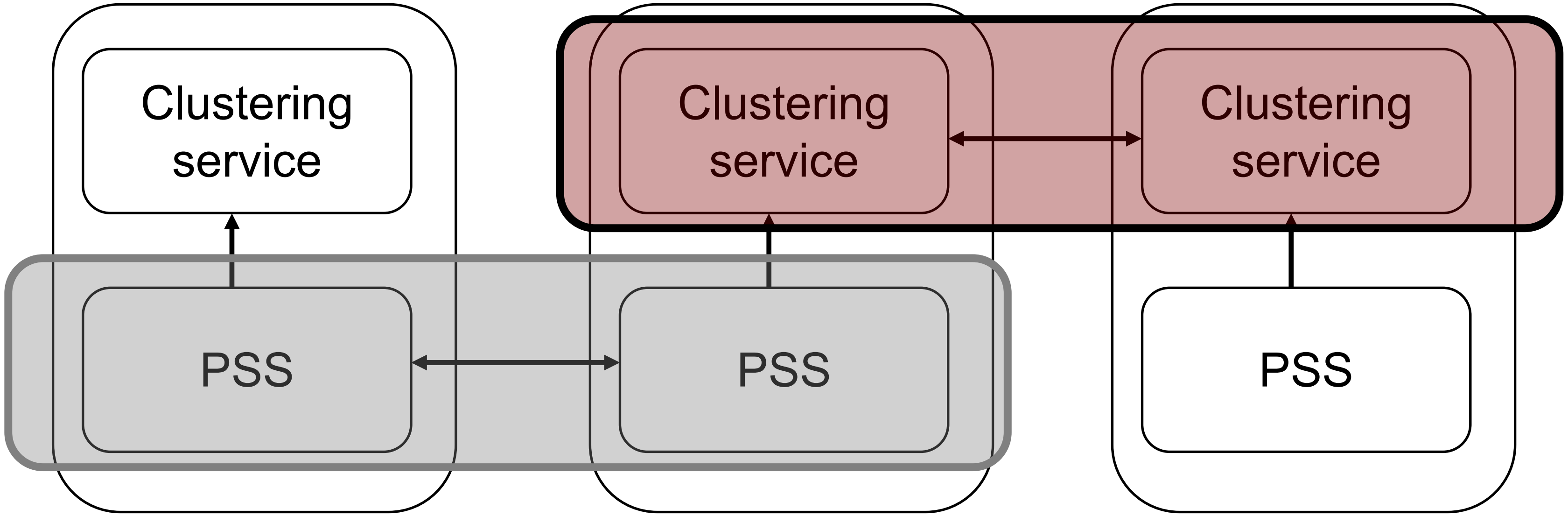
# System model

- Semantic view of *l* semantic neighbours
- Semantic proximity function *S(P,Q).*
  - The higher the value of *S(P,Q),* the “closer” the nodes.
  - The objective is to fill P’s semantic view to optimize

$$\sum_{i=1}^{l} S(P,Q_i)$$

# Gossiping framework

- Target selection
  - Close peers
  - All nodes are examined: create a “small-world” like structure so that new nodes are discovered.

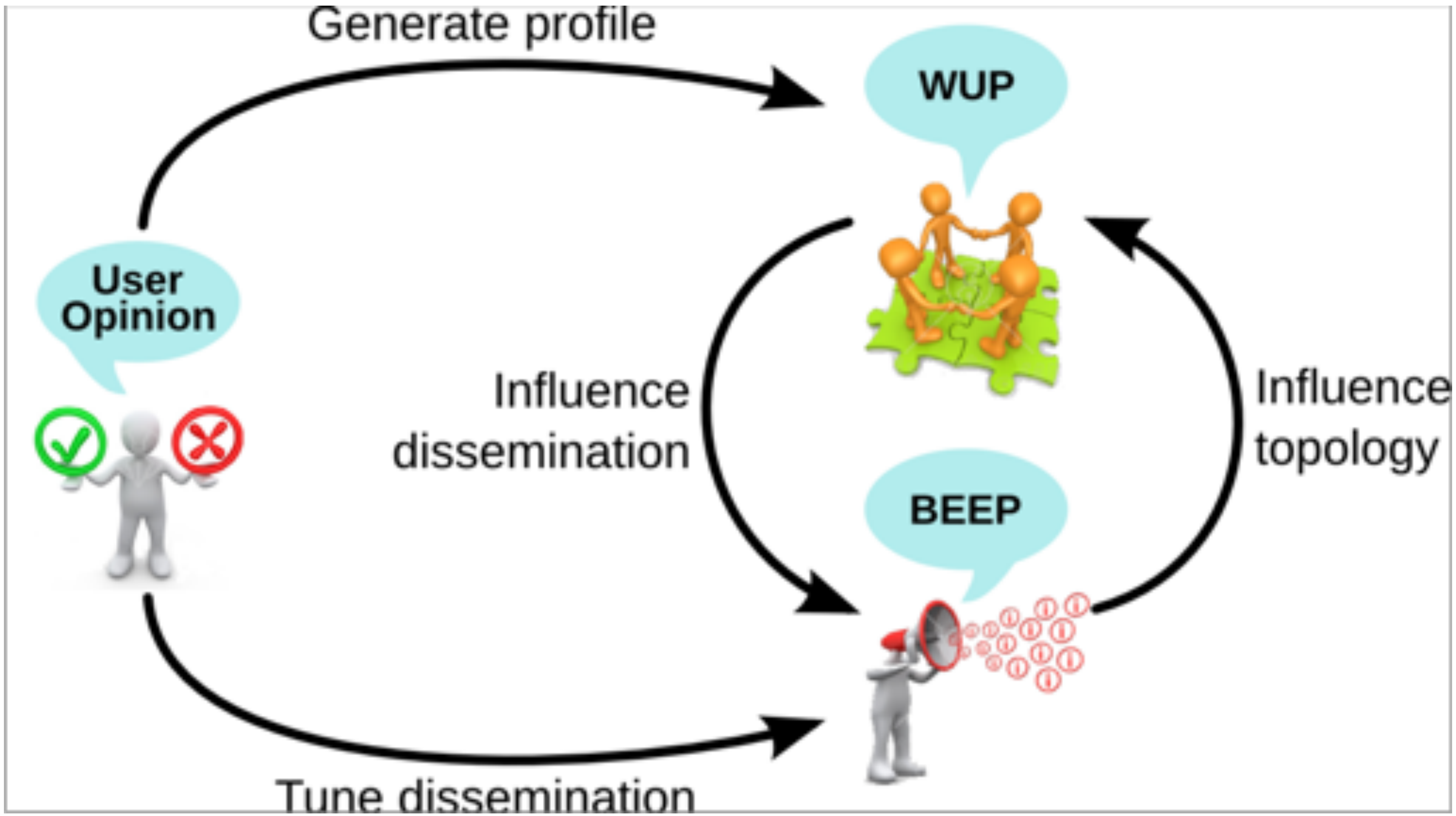


# Outline

- Decentralized Recommendation - > WhatsUP
- Privacy by Profile Blurring
- Privacy by Proxy
- Privacy through Landmarks

Antoine Boutet, Davide Frey, Rachid Guerraoui, Arnaud Jégou, Anne-Marie Kermarrec:
**WHATSUP: A Decentralized Instant News Recommender.** IPDPS 2013: 741-752

# WhatsUp in a nutshell



# WhatsUp challenges

Who are my social acquaintances ⇒ Similarity metric

How to discover them? ⇒ Sampling

How to disseminate news items? ⇒ Biased epidemic protocol

How to preserve users’privacy

# Which nodes for the social network?

**Model**

*U*(sers) × *I*(tems) (news items)

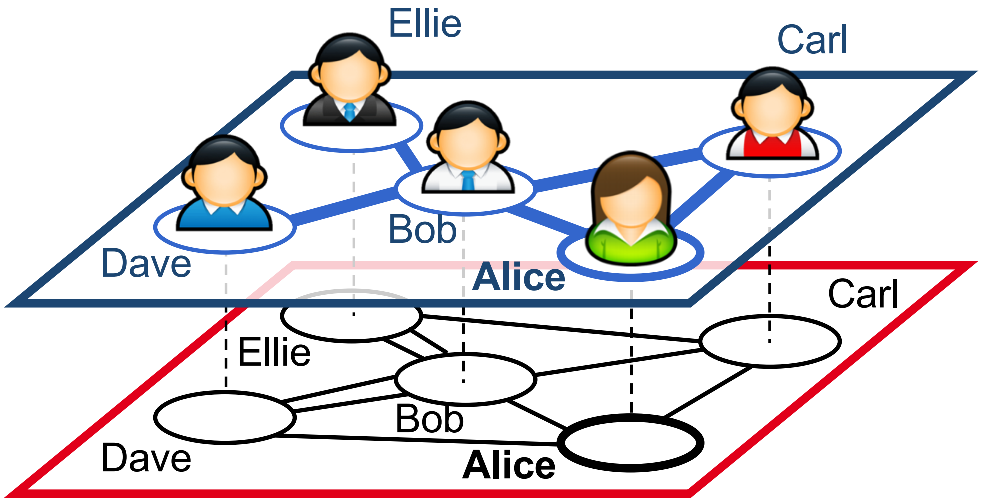
*Profile(u) = vector of liked news items*

**Cosine similarity metric**

$$Similarity\ (n,p) = \frac{n.p}{||n||\ ||p||}$$

Minimal information: **no tag, no user's input**

# The WhatsUp social network



**clustering** layer
gossip-based
topology clustering

**RPS** layer providing
random sampling

— *Random link*

▬ *Social link*

○ *node*

# Clustering through Similarity

Similarity evaluates the closeness of two vectors, A and B, representing profiles.

Overlap is not enough -> cosine similarity

Generic vectors

$$Cos = \frac{A \cdot B}{||A||||B||}$$

Binary vectors

$$Cos = \frac{A \cap B}{\sqrt{|A||B|}}$$

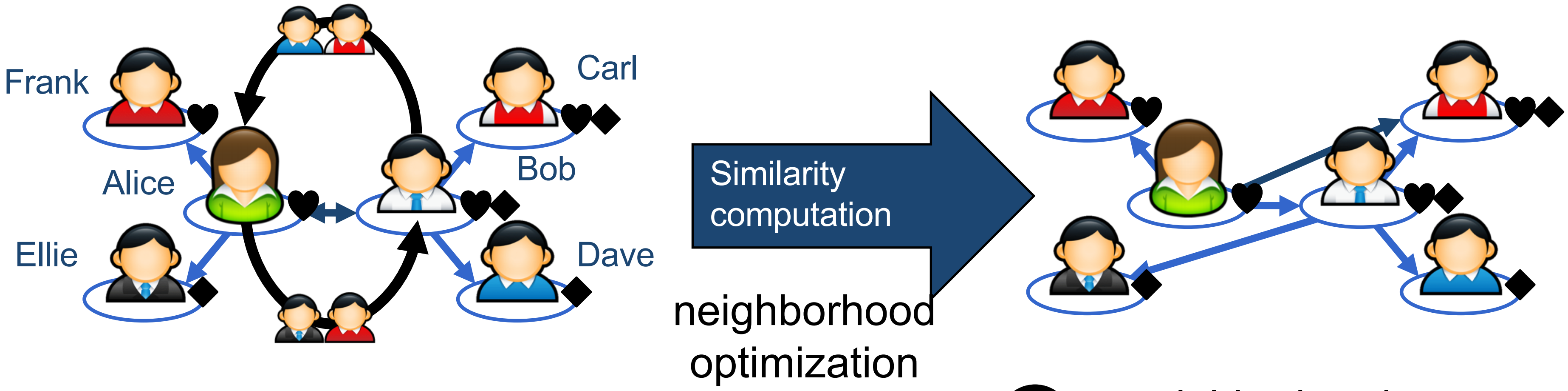
WUP Similarity

$$Wup = \frac{sub(A,B) \cdot B}{||A||||B||}$$

$$Wup = \frac{sub(A,B) \cap B}{\sqrt{|A||B|}}$$

$sub(A,B) =$ Scores in A for items that exist in B

# Model: P2P similarity-based network



Similarity computation

neighborhood optimization

❶ exchange of neighbors lists

❷ neighborhood optimization

# Data structures

**Social Network of the c closest entries**

| | |
|---|---|
| **@IP:port** | 132.154.8.5:2020 |
| **Bloom Filter** | 010111011001 |
| **Profile** | I like it: : $N_{1,}$ $N_{2,}$ …<br>I don't : $N_{10'}$ $N_{13,}$<br>… |
| **Update time** | 5 |

Exchange of Bloom filters

**Uniform (dynamic) sample of k random entries**

| | | |
|---|---|---|
| **@IP: port** | 102.14.18 .1:2110 | |
| **Bloom Filter** | 10010000 0110 | |
| **Update time** | 30 | |

# WhatsUp challenges

Who are my social acquaintances

How to discover them?

**How to disseminate news items ?** → Biased epidemic protocol (BEEP)

# BEEP: orientation and amplification

Orientation: **to whom?**

Forward to friends

Forward to random users

Amplification: **to how many?**

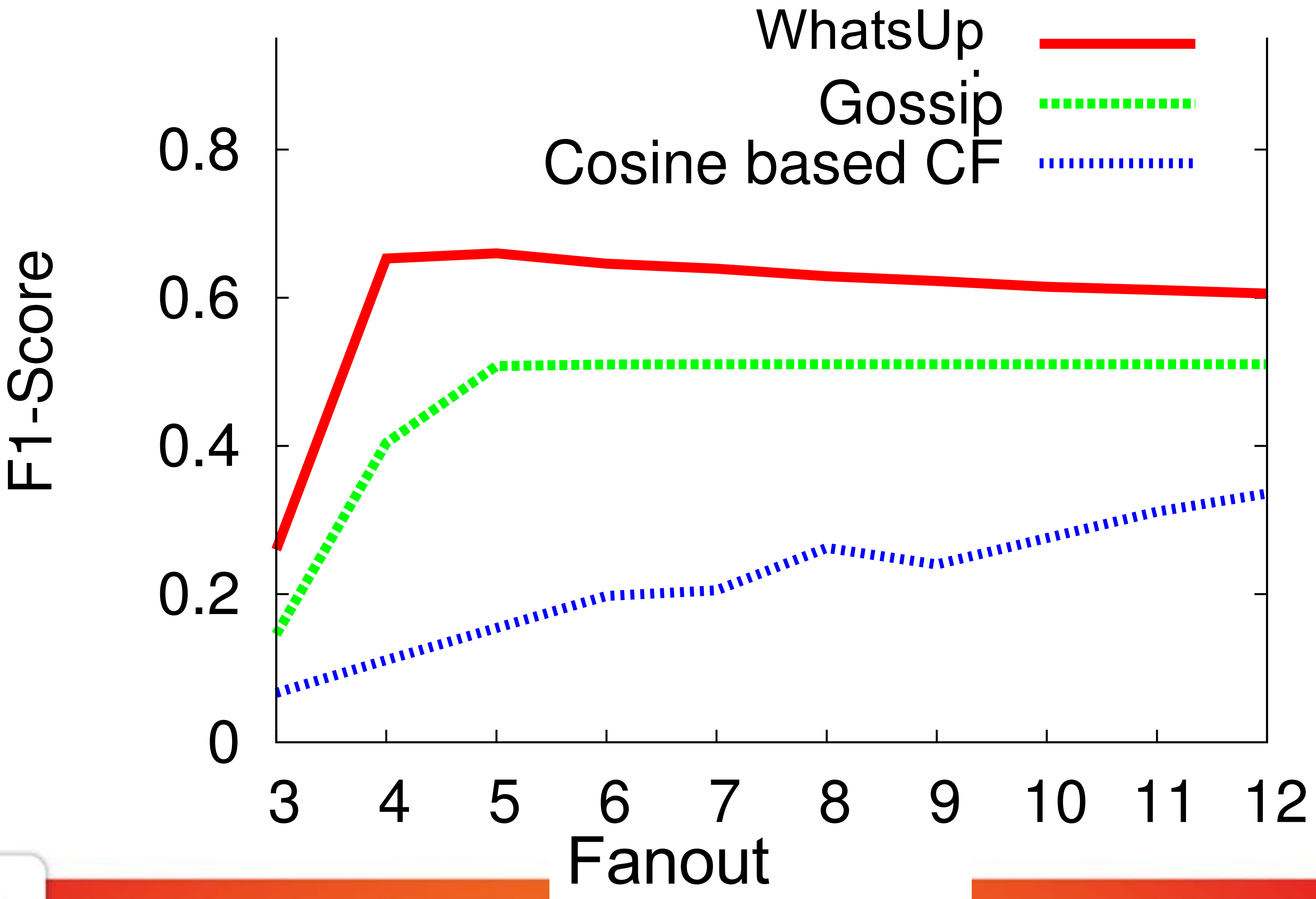
Increase fanout

Decrease fanout

# WhatsUp in action on the survey

| | **Precision** | **Recall** | **Redundancy** | **Messages** |
|---|---|---|---|---|
| Gossip | 0.34 | 0.99 | 0.85 | 2.3 M |
| Cosine-CF | 0.64 | 0.12 | 0.27 | 30k |
| **Whatsup** | **0.53** | **0.78** | **0.28** | **280k** |

# WhatsUp in action



# WhatsUp challenges

Who are my social acquaintances?

How to discover them?

How to disseminate news items ?

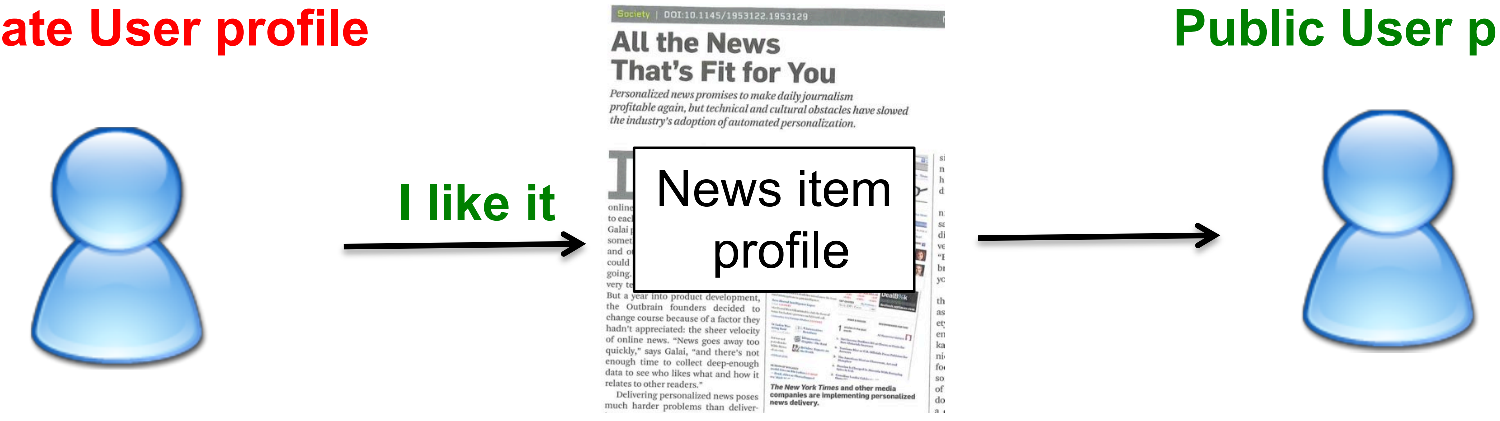
**How to preserve users' privacy?**

# Outline

- Decentralized Recommendation
- Privacy by Profile Blurring -> Compact Profiles
- Privacy by Proxy
- Privacy through Landmarks

Antoine Boutet, Davide Frey, Rachid Guerraoui, Arnaud Jégou, Anne-Marie Kermarrec:
**Privacy-Preserving Distributed Collaborative Filtering.** NETYS 2014: 169-184

# Privacy by Profile Blurring



**Private User profile**

**I like it**

Society | DOI:10.1145/1953122.1953129

All the News That's Fit for You

*Personalized news promises to make daily journalism profitable again, but technical and cultural obstacles have slowed the industry's adoption of automated personalization.*

But a year into product development, the Outbrain founders decided to change course because of a factor they hadn't appreciated: the sheer velocity of online news. "News goes away too quickly," says Galai, "and there's not enough time to collect deep-enough data to see who likes what and how it relates to other readers."

Delivering personalized news poses much harder problems than deliver-

***The New York Times* and other media companies are implementing personalized news delivery.**

News item profile

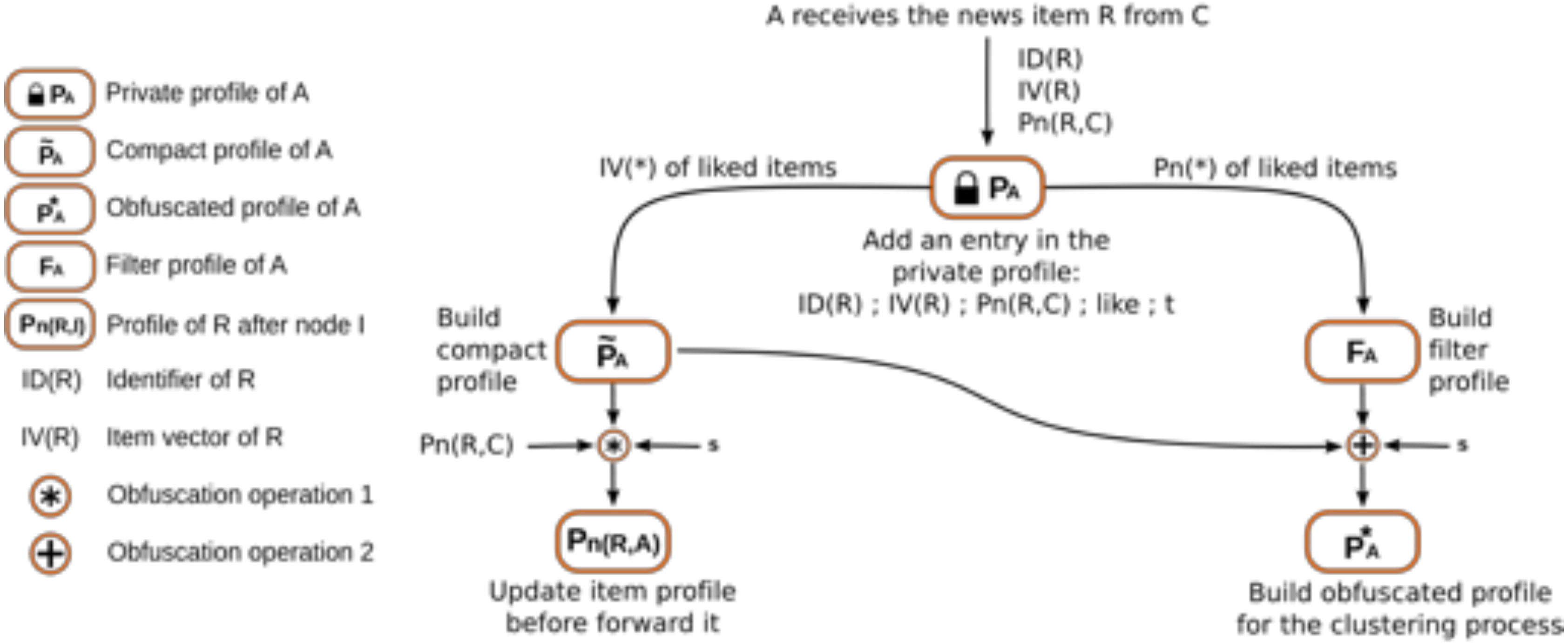
**Public User profile**

User Profile
used locally for
similarity computation

Aggregation of profiles
of users who liked
the news item

User Profile
exchanged
during gossip

# Privacy by Profile Blurring



| | |
|---|---|
| 🔒 $P_A$ | Private profile of A |
| $\tilde{P}_A$ | Compact profile of A |
| $P^*_A$ | Obfuscated profile of A |
| $F_A$ | Filter profile of A |
| $P_{n(R,I)}$ | Profile of R after node I |
| ID(R) | Identifier of R |
| IV(R) | Item vector of R |
| ⊛ | Obfuscation operation 1 |
| ⊕ | Obfuscation operation 2 |

A receives the news item R from C

ID(R)
IV(R)
Pn(R,C)

IV(*) of liked items

Pn(*) of liked items

Add an entry in the private profile:
ID(R) ; IV(R) ; Pn(R,C) ; like ; t

Build compact profile

Build filter profile

Pn(R,C)

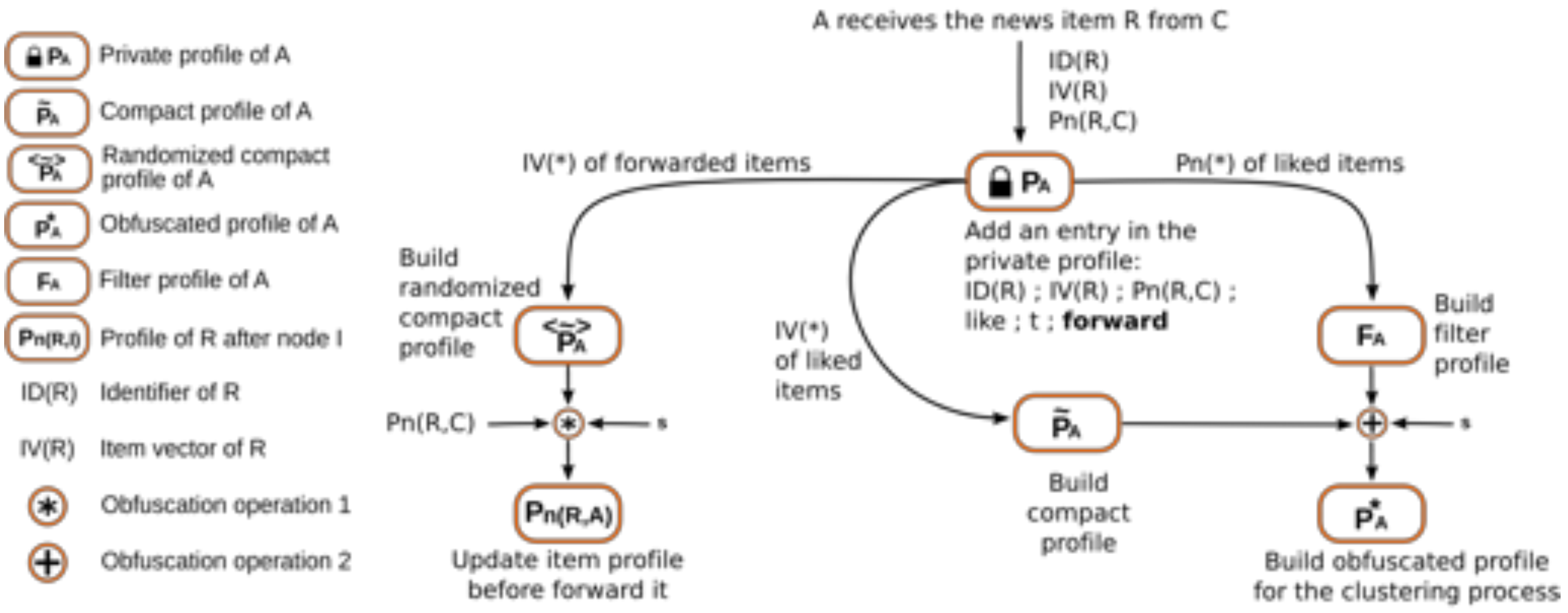
s

s

Update item profile before forward it

Build obfuscated profile for the clustering process

# Private Dissemination



🔒 $P_A$ Private profile of A

$\tilde{P}_A$ Compact profile of A

$\overset{<\sim>}{P}_A$ Randomized compact profile of A

$P^*_A$ Obfuscated profile of A

$F_A$ Filter profile of A

$P_{n(R,I)}$ Profile of R after node I

ID(R) Identifier of R

IV(R) Item vector of R

⊛ Obfuscation operation 1

⊕ Obfuscation operation 2

A receives the news item R from C

ID(R)
IV(R)
Pn(R,C)

🔒 $P_A$

Add an entry in the private profile:
ID(R) ; IV(R) ; Pn(R,C) ;
like ; t ; **forward**

IV(*) of forwarded items

Pn(*) of liked items

IV(*) of liked items

Build randomized compact profile

$\overset{<\sim>}{P}_A$

Pn(R,C) → ⊛ ← s

$P_{n(R,A)}$

Update item profile before forward it

$\tilde{P}_A$

Build compact profile

$F_A$

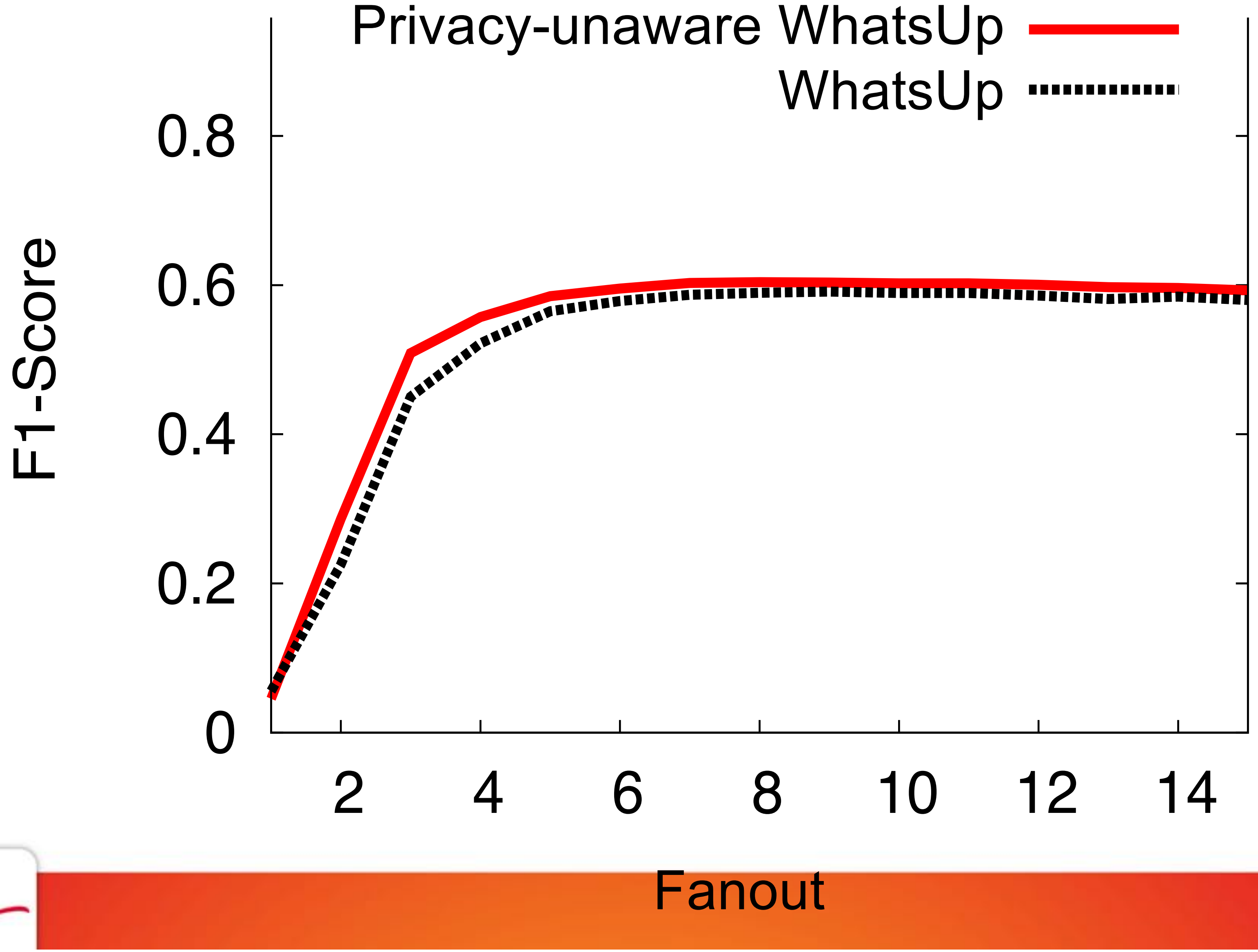
Build filter profile

⊕ ← s

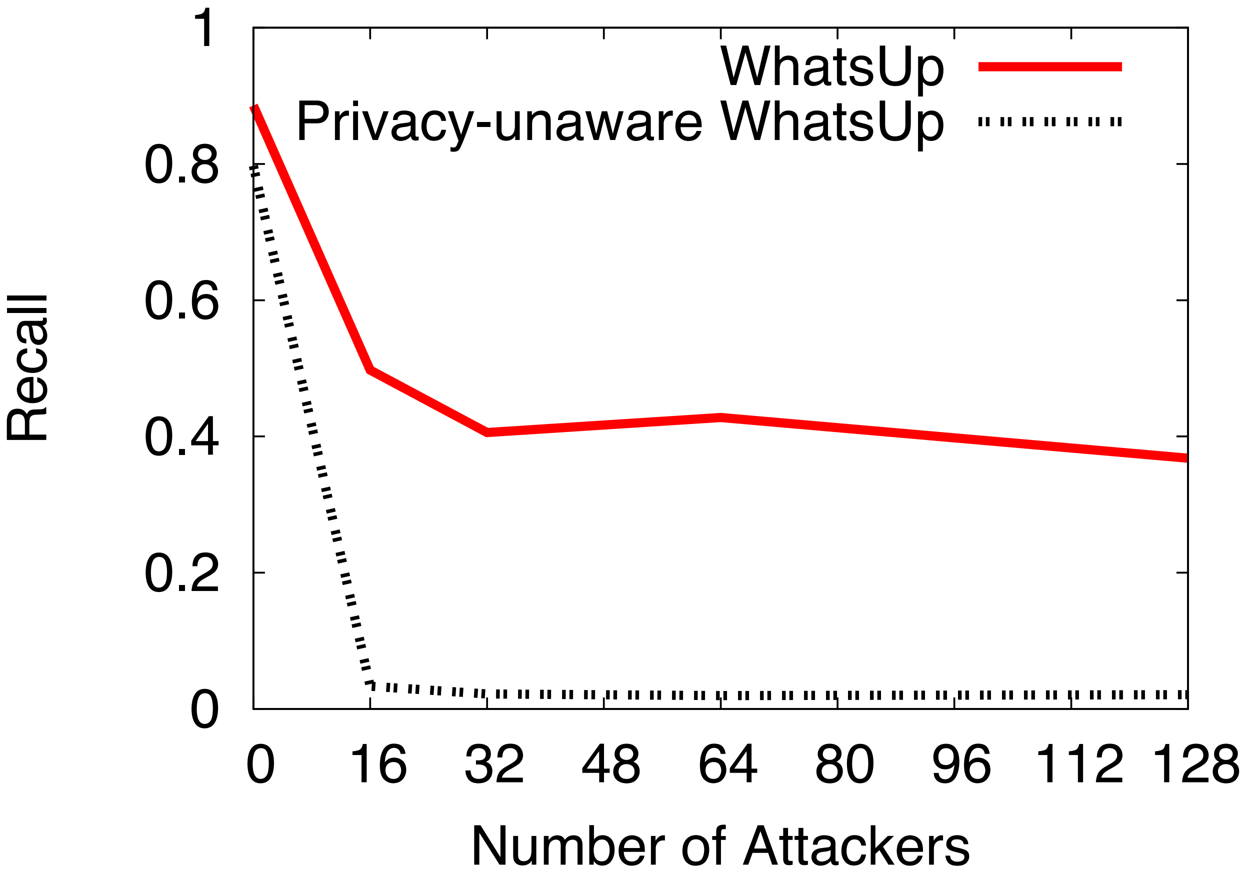
$P^*_A$

Build obfuscated profile for the clustering process

# Impact of profile bluring



# Resilience to attacks



# Outline

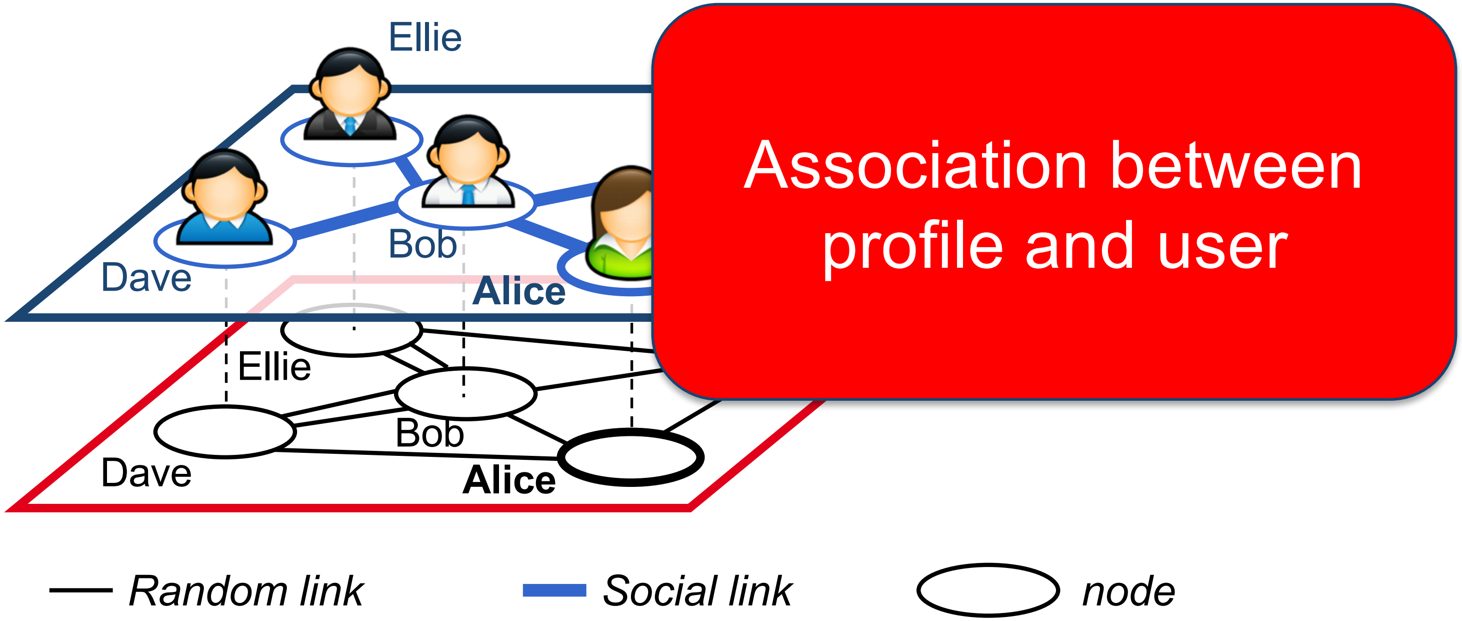
- Decentralized Recommendation
- Privacy by Profile Blurring
- Privacy by Proxy -> FreeRec
- Privacy through Landmarks

Antoine Boutet, Davide Frey, Arnaud Jégou, Anne-Marie Kermarrec, Heverson B. Ribeiro:
**FreeRec: an anonymous and distributed personalization architecture.** Computing 97(9): 961-980 (2015)

# Privacy through Anonymity

**Clustering** layer
gossip-based
topology clustering

**RPS** layer providing
random sampling

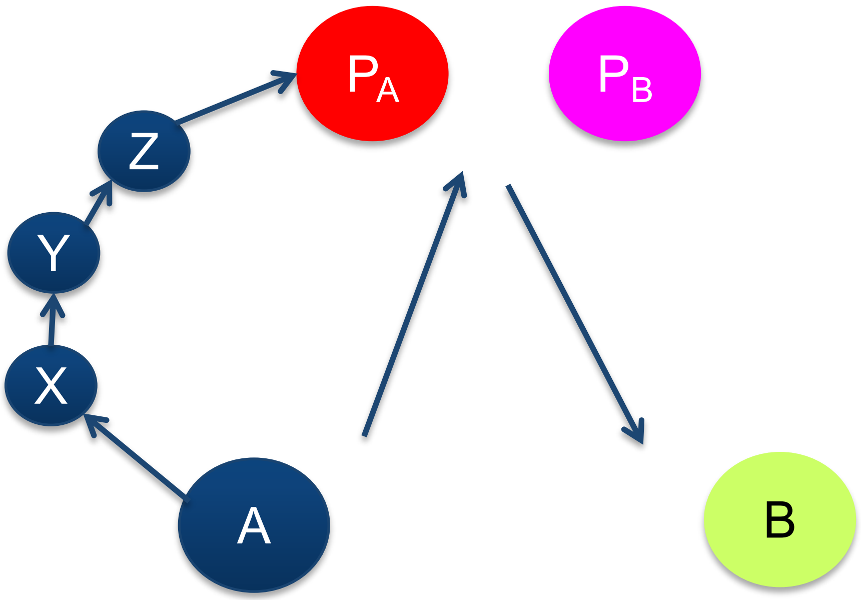


Association between profile and user

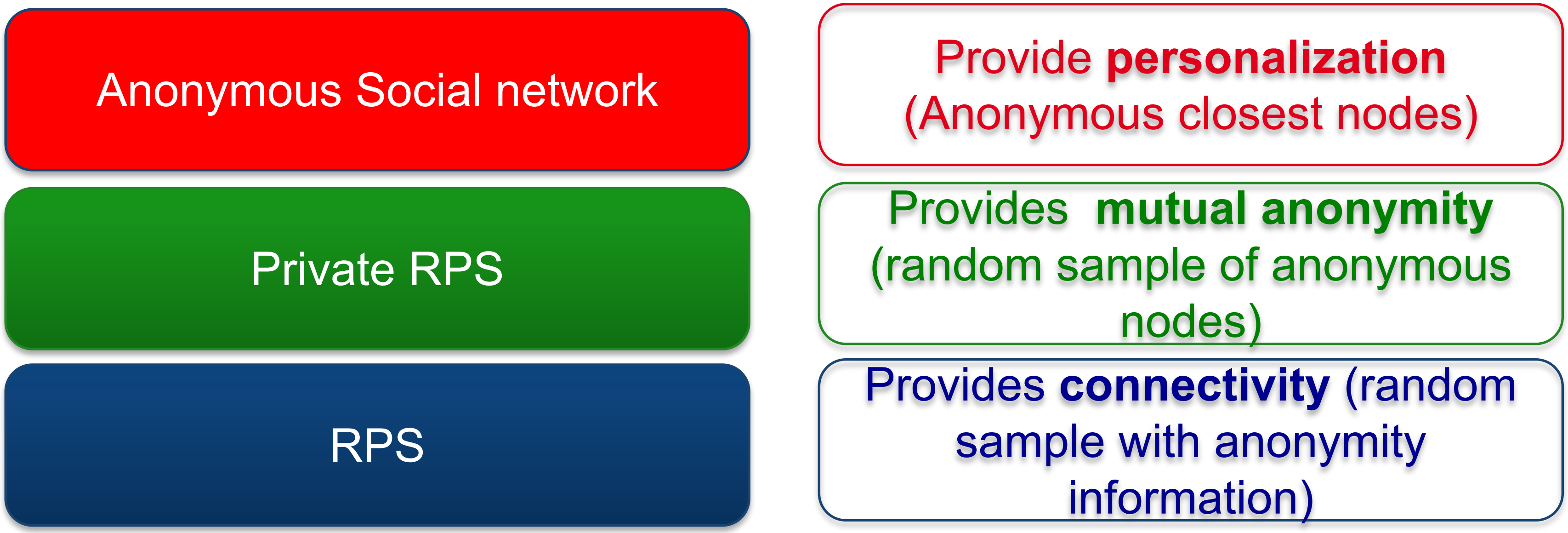
# Onion-like proxy chain

**Dissociates the profile from the user's identifier**

**User's pseudo = IP@of its proxy**



# FreeRec architecture



Anonymous Social network

Provide **personalization** (Anonymous closest nodes)

Private RPS

Provides **mutual anonymity** (random sample of anonymous nodes)

RPS

Provides **connectivity** (random sample with anonymity information)

**Adapt to churn (node arrival and departure)**
**Evaluated on simulation and PlanetLab deployement**

# Data Structures

Message key

Public Chain key : stored in RPS

Secret key

Chain Table

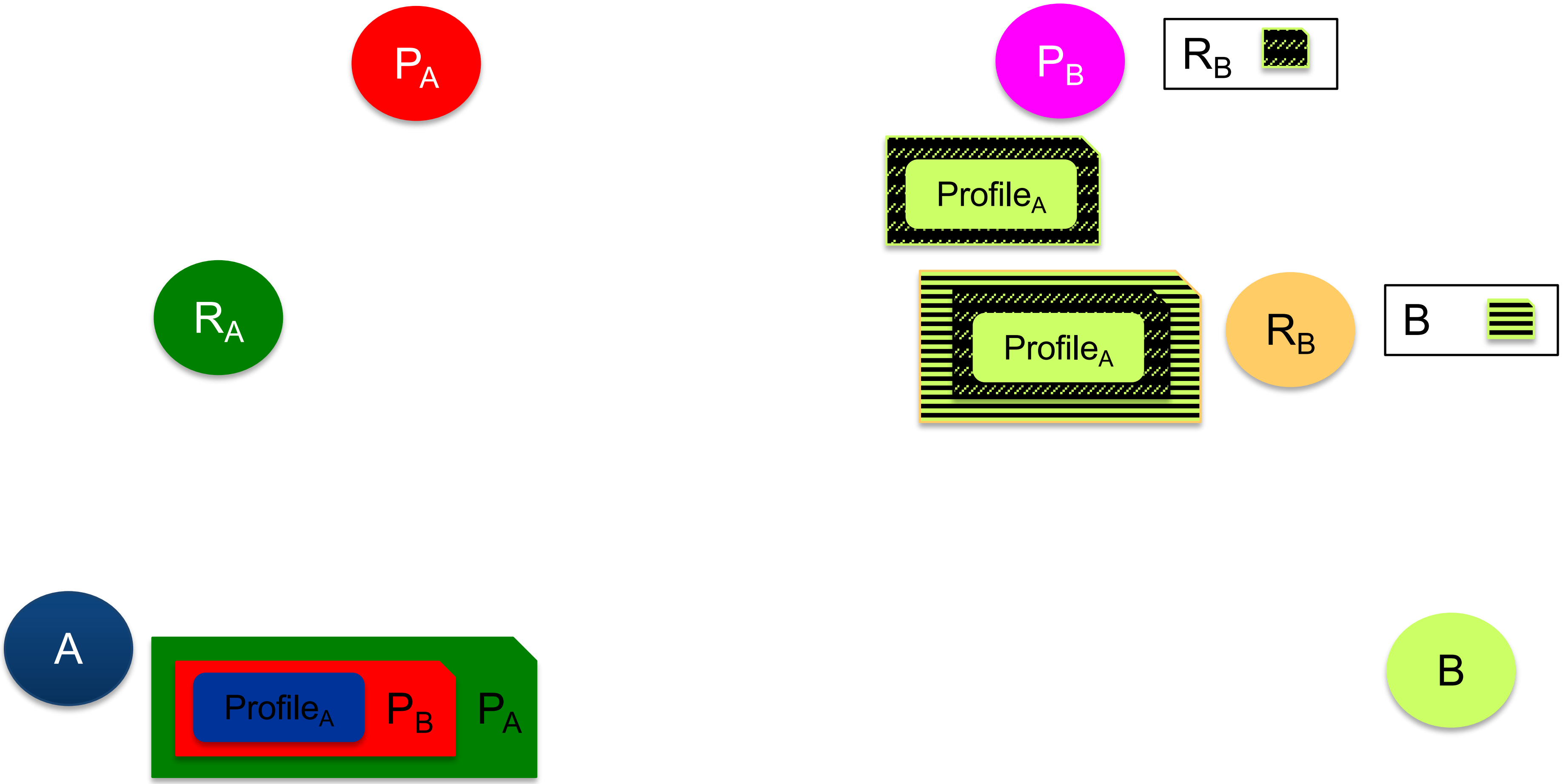
Routing Table: store routingIds

RPS: IP@ + chain key, no profile

PRPS: entry for b is (proxy $p_b$)

- $p_b$'s RoutingId
- $p_b$ ‘s IP@
- $p_b$'s public chain key
- b’s public message key
- b’s profile

# Anonymous Profile exchange in FreeRec



# EXPERIMENTS

# Experimental setup

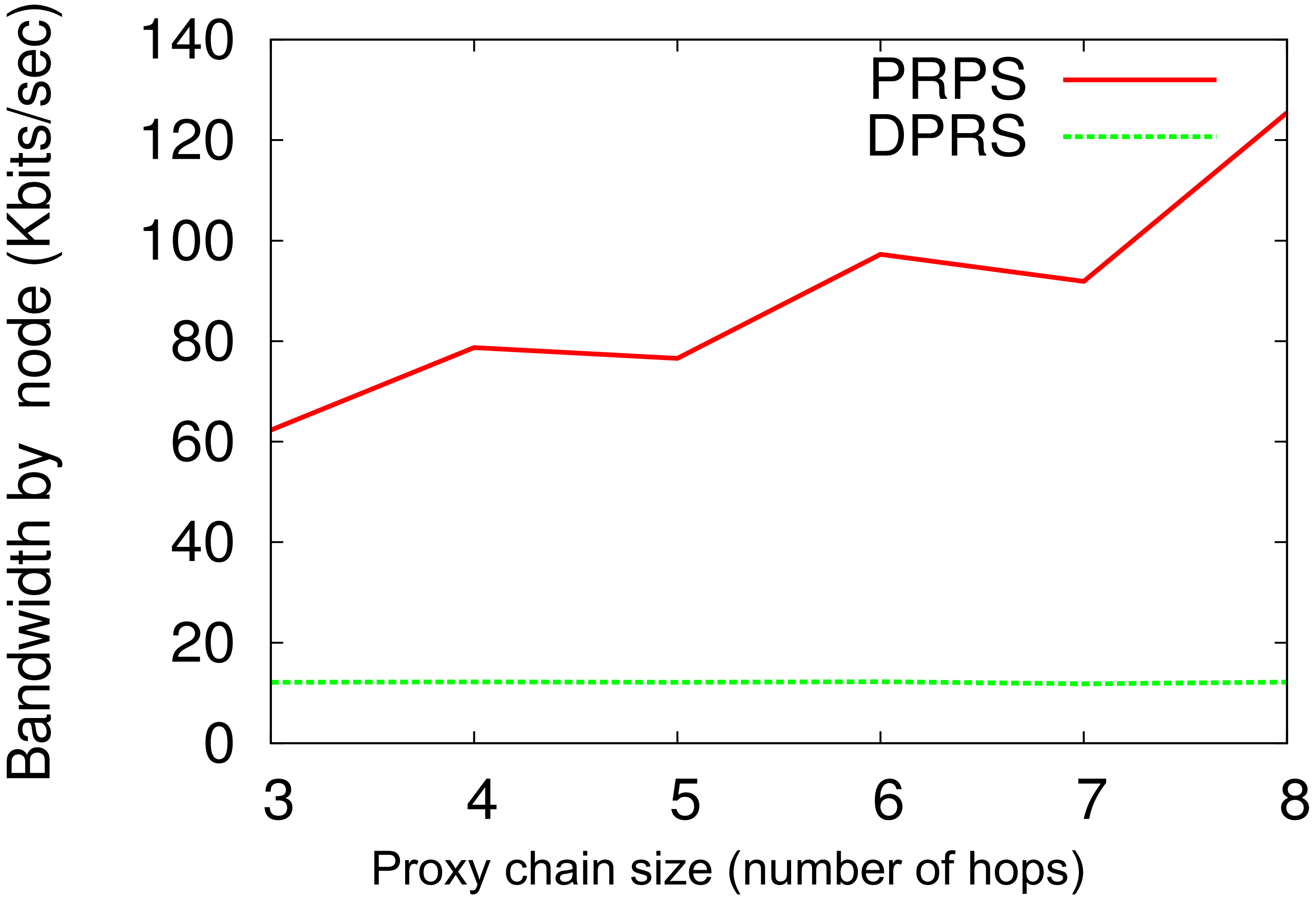
**System metrics**:

- Simulations: Overhead (traffic), Message loss, Number of hops
- PlanetLab: bandwidth and latency

**User Metrics**: Recall-Precision

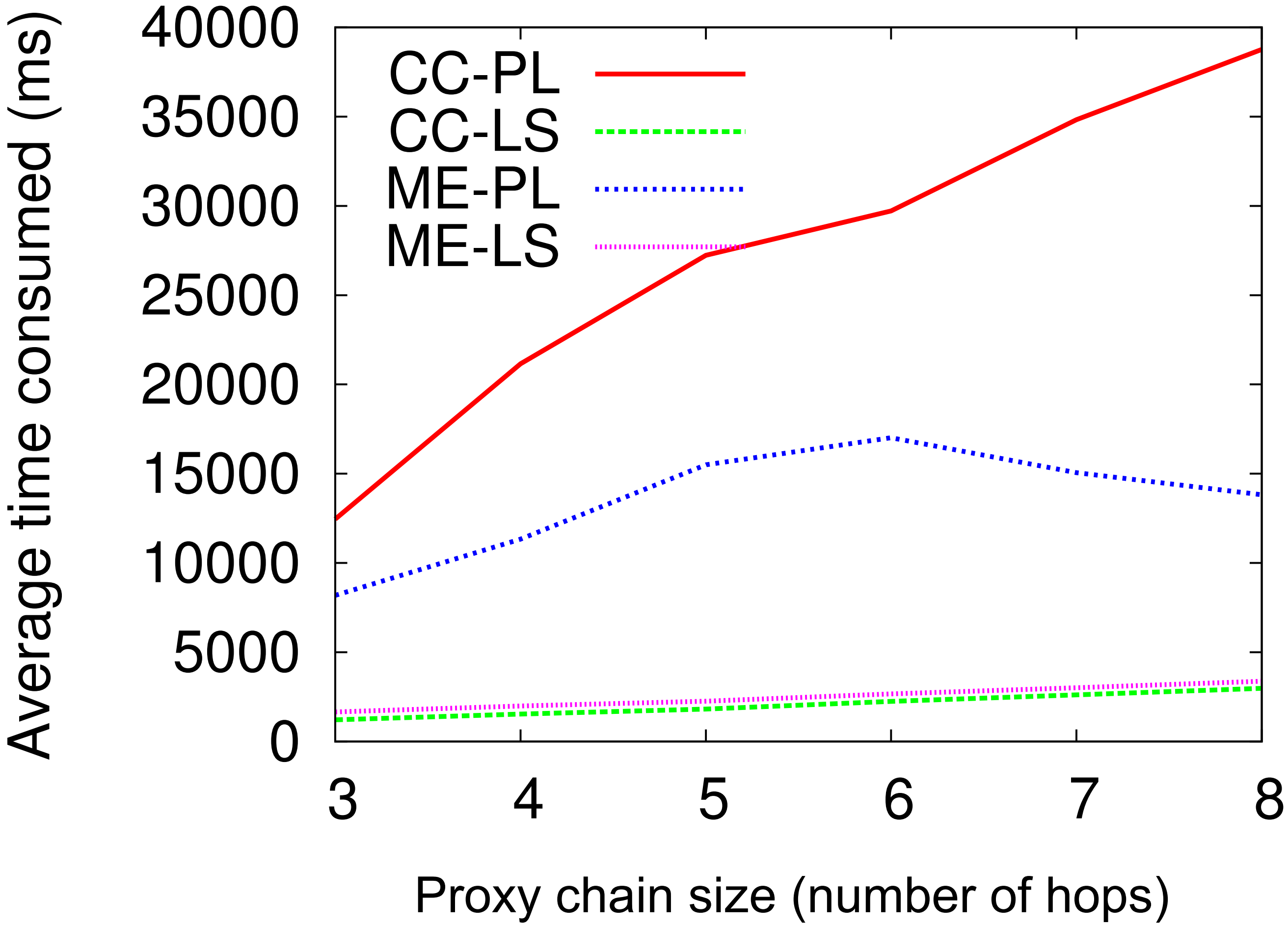
**Dataset:** Real survey, 535 users on 1235 news items

# Overhead

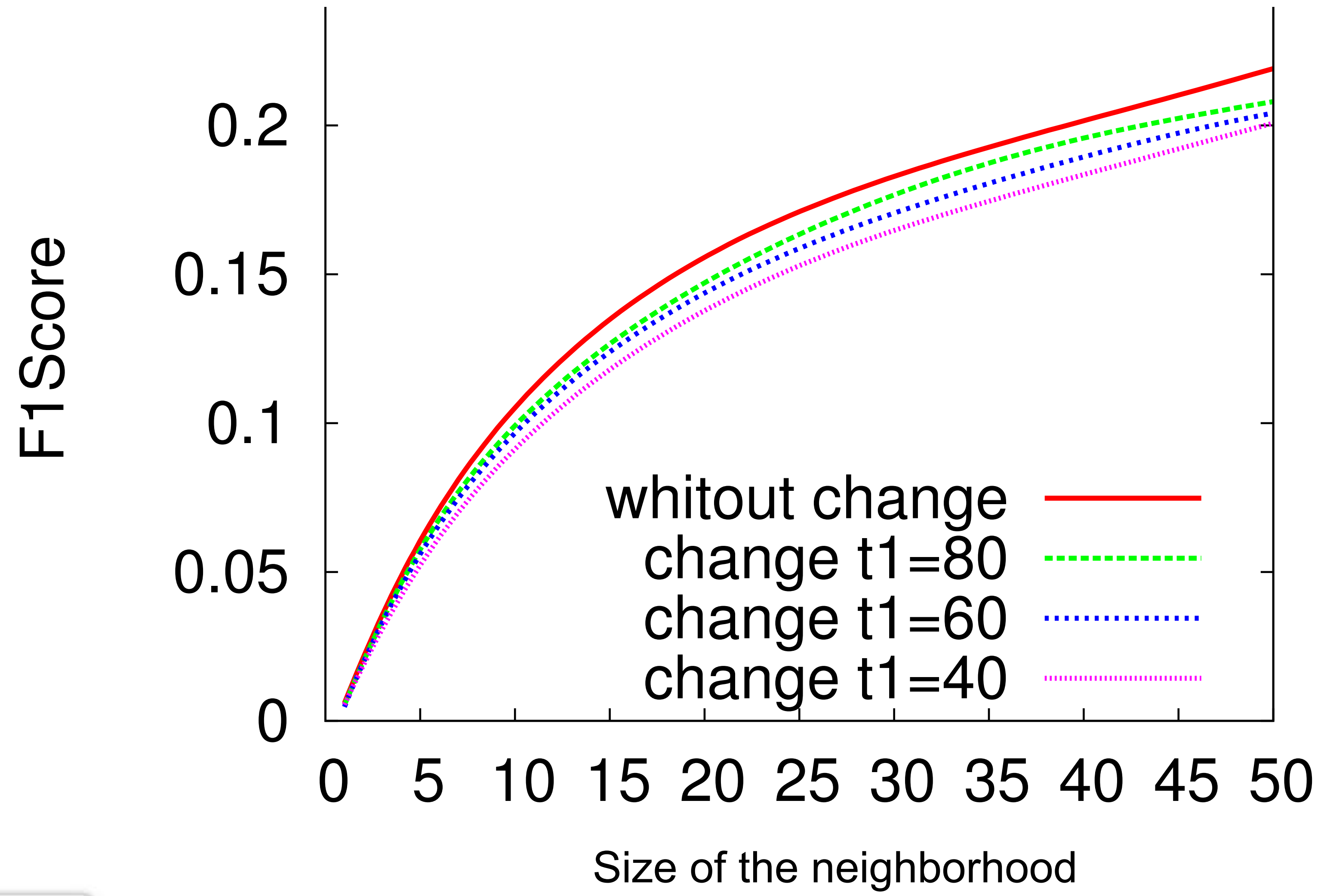


Bandwidth by node (Kbits/sec)
140
120
100
80
60
40
20
0
3
4
5
6
7
8
Proxy chain size (number of hops)
PRPS
DPRS

# Latency (in ms)



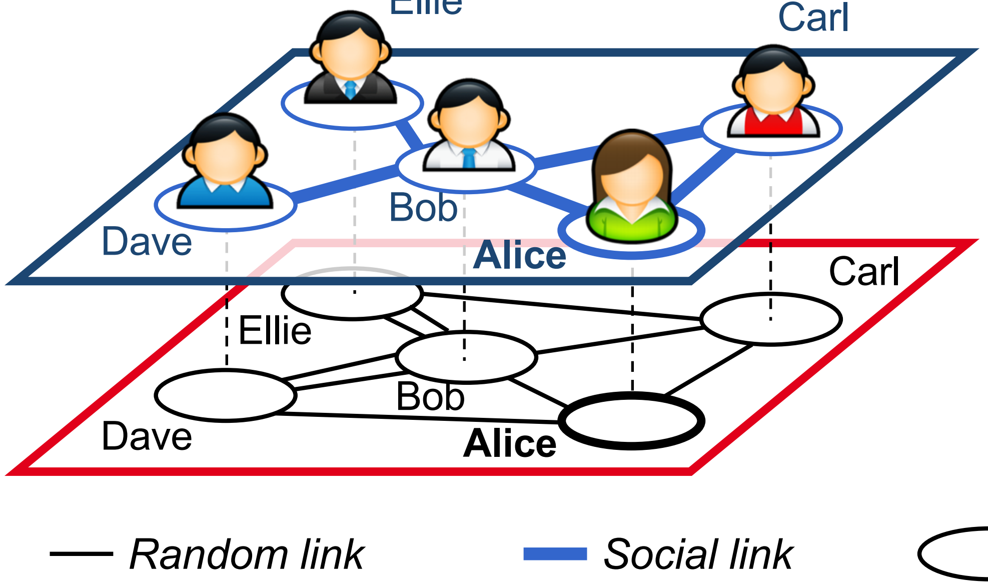
# Impact on message loss: change of proxy chain



# Outline

- Decentralized Recommendation
- Privacy by Profile Blurring
- Privacy by Proxy
- Privacy through Landmarks -> Hide&Share

# Peer-to-Peer Collaborative Filtering



**clustering** layer
gossip-based
topology clustering

**RPS** layer providing
random sampling

- Remove Big Brother

# Peer-to-Peer Collaborative Filtering

Build Knn graph through epidemic protocols

- RPS builds a random topology
- Continuously provides new information
- Clustering identifies nearest neighbors
- Similarity metric: e.g. cosine
- Recommendation based on neighbors' ratings

# Key Privacy Leak: Similarity Computation

Computing similarities requires

knowledge of each other's profiles

Replace big brother by many little brothers

# Attacker Model

- Goal: Discover a target user’s interests
- Restricted active adversary
- Passive information gathering □
- Some active steps:
- Tap unencrypted communications
- Try to bias multi-party computations
- Unlimited similarity computations
- □ No collusion, no Sybil attack

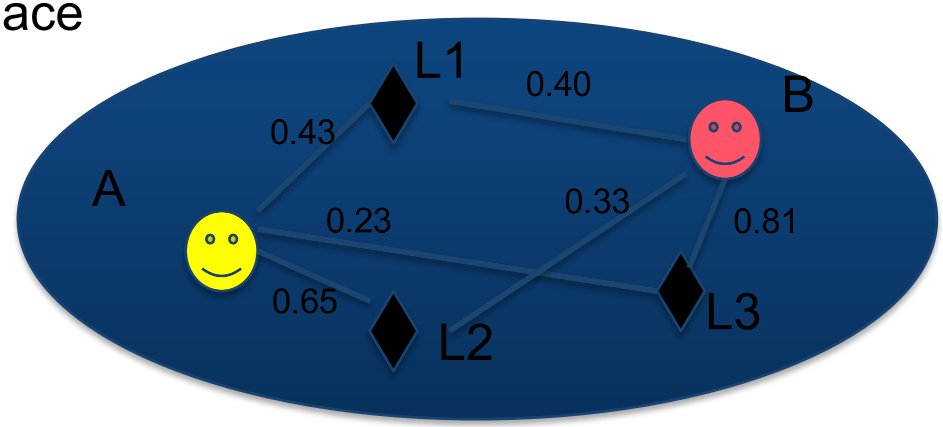
# Hide and Share

Main Insight: Landmark-based similarity

- Indirectly compare user profiles by exploiting their similarities

with randomly generated profiles (landmarks)

Profile space

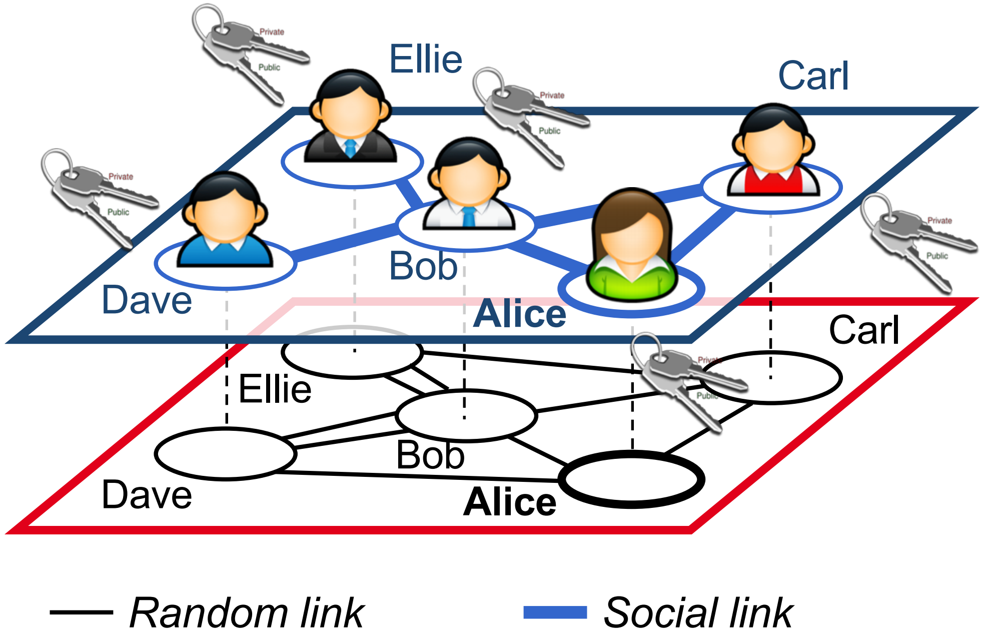


Coordinate system analogy

# Hide and Share Requirements

- Computation Confidentiality
- Landmark-profile independence
- Fair Landmark generation
- Time-independent information release

# Computation confidentiality



Attach Public Key to gossip messages

Generate secret key to exchange data for similarity computation

# Landmark-profile Independence

•Need to generate random landmarks

•Need a way to describe the profile space!

- Represent profiles as binary vectors
  - Profile is a set of items
  - Compact profile in the form of bloom filters
    - Only count “liked” items (rating>threshold)

# Fair Landmark Generation

- Need common seed
- Bit-commitment – blum’s protocol

P1 and P2 flip a coin
P1 sends f(conc(result, nonce))
P2 reveals result to P1
P1 reveals result to P1
If same result -> bit = 1

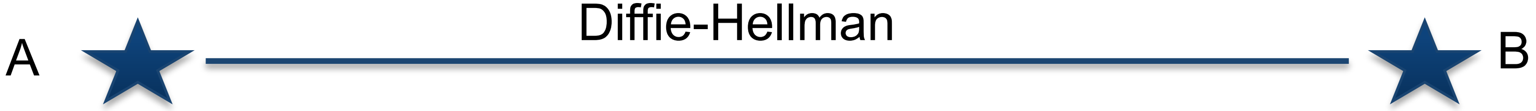
# Time-independent information release

- Generate landmarks using common seed
- Store seed for future use
- Will recompute the same landmarks the next time it meets peer.
- Overhead -> one seed per peer

# Protocol Summary

## A and B's first meeting

Set up secure communication channel



# Protocol Summary

## A and B’s first meeting

Set up secure communication channel

Agree on common seed

Bit commitment -> seed
A
B

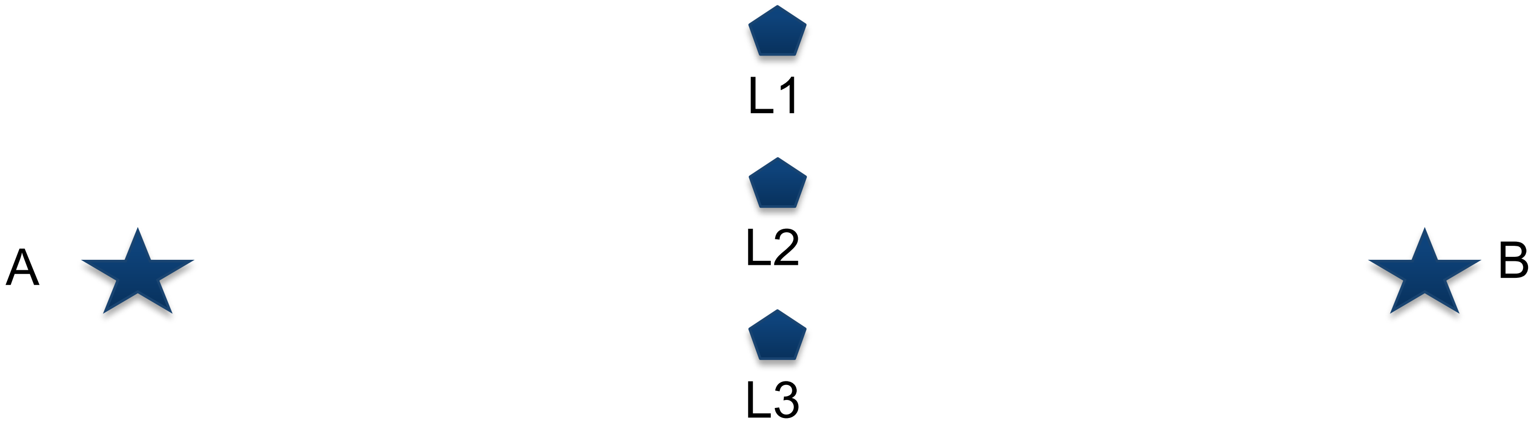
# Protocol Summary

## A and B’s first meeting

Set up secure communication channel

Agree on common seed

Derive L random profiles (landmarks) using the seed



# Protocol Summary

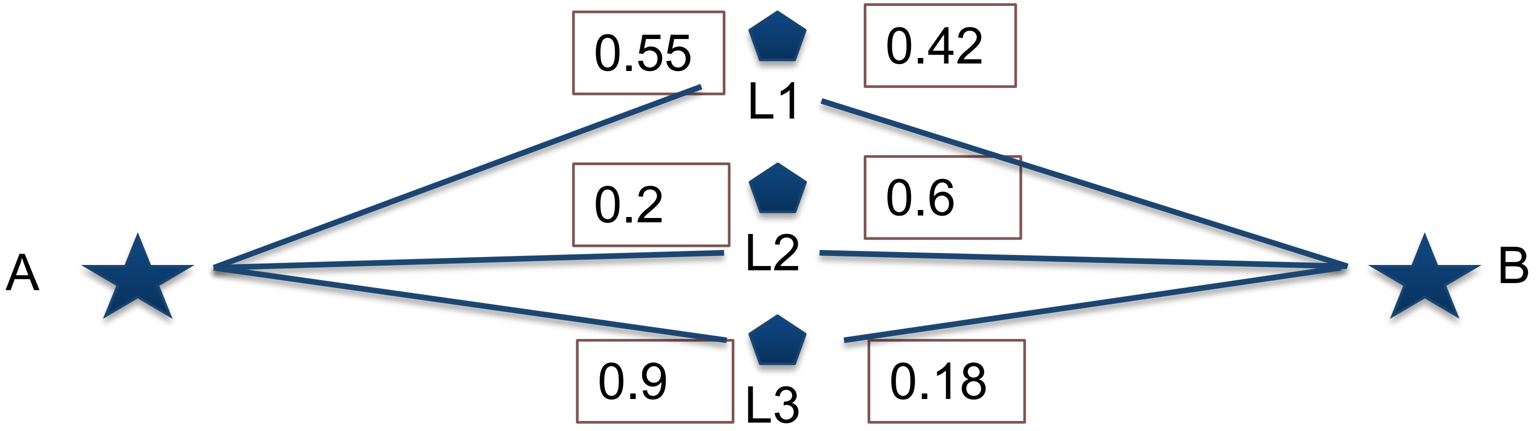
## A and B's first meeting

Set up secure communication channel

Agree on common seed

Derive L random profiles (landmarks) using the seed

Compute similarity with the landmarks



# Protocol Summary

## A and B's first meeting

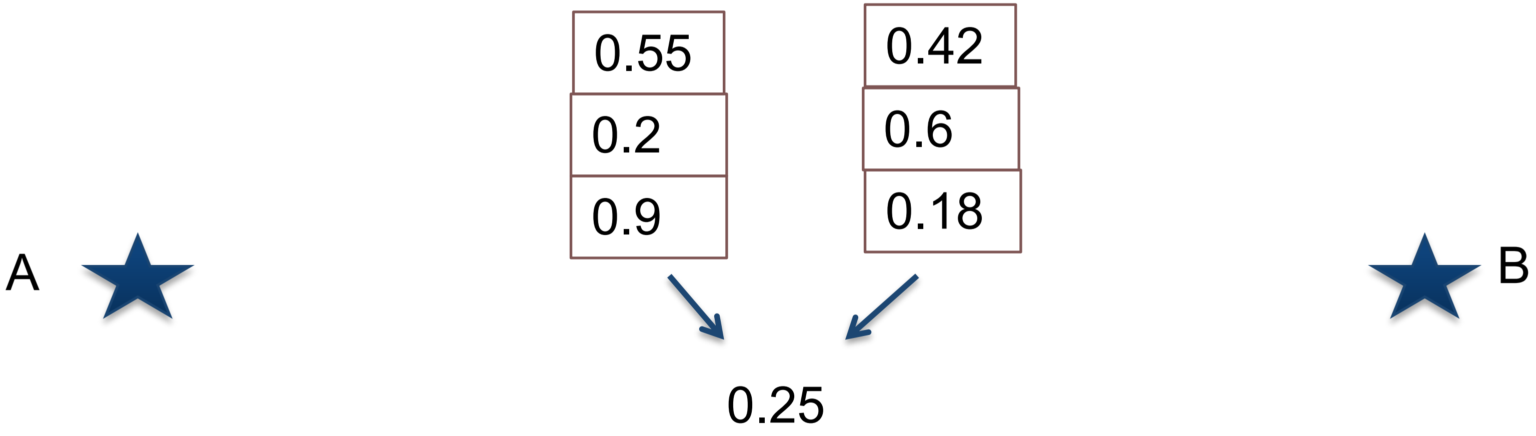
Set up secure communication channel

Agree on common seed

Derive L random profiles (landmarks) using the seed

Compute similarity with the landmarks

Cosine similarity of coordinate vectors



A

0.55
0.2
0.9

0.42
0.6
0.18

B

0.25

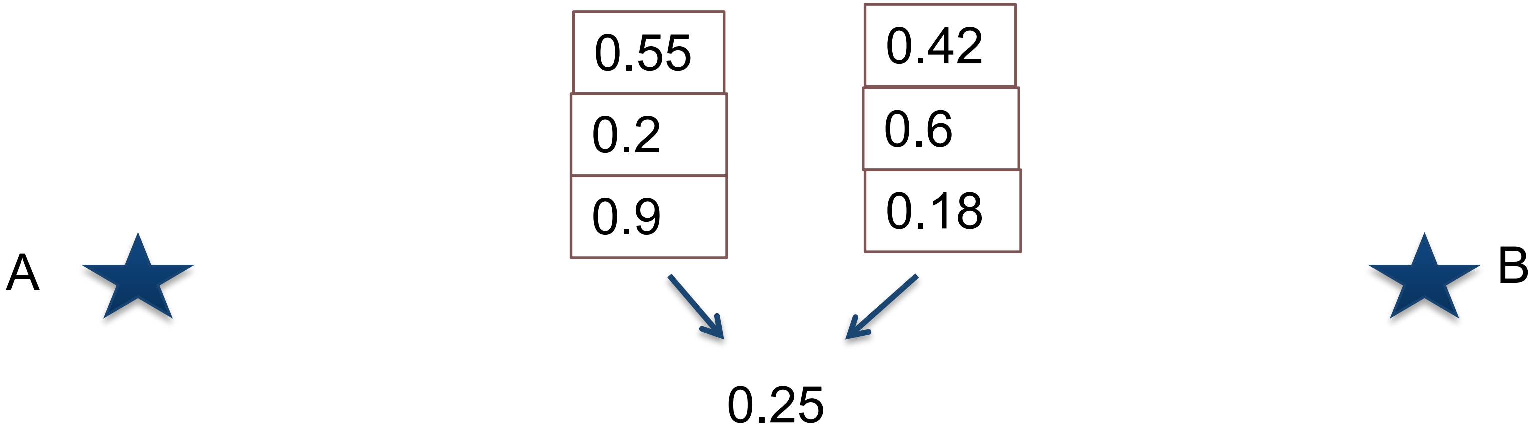
# Protocol Summary

A and B meet again

Derive L random profiles (landmarks) using the seed

Compute similarity with the landmarks

Cosine similarity of coordinate vectors



A

0.55

0.2

0.9

0.42

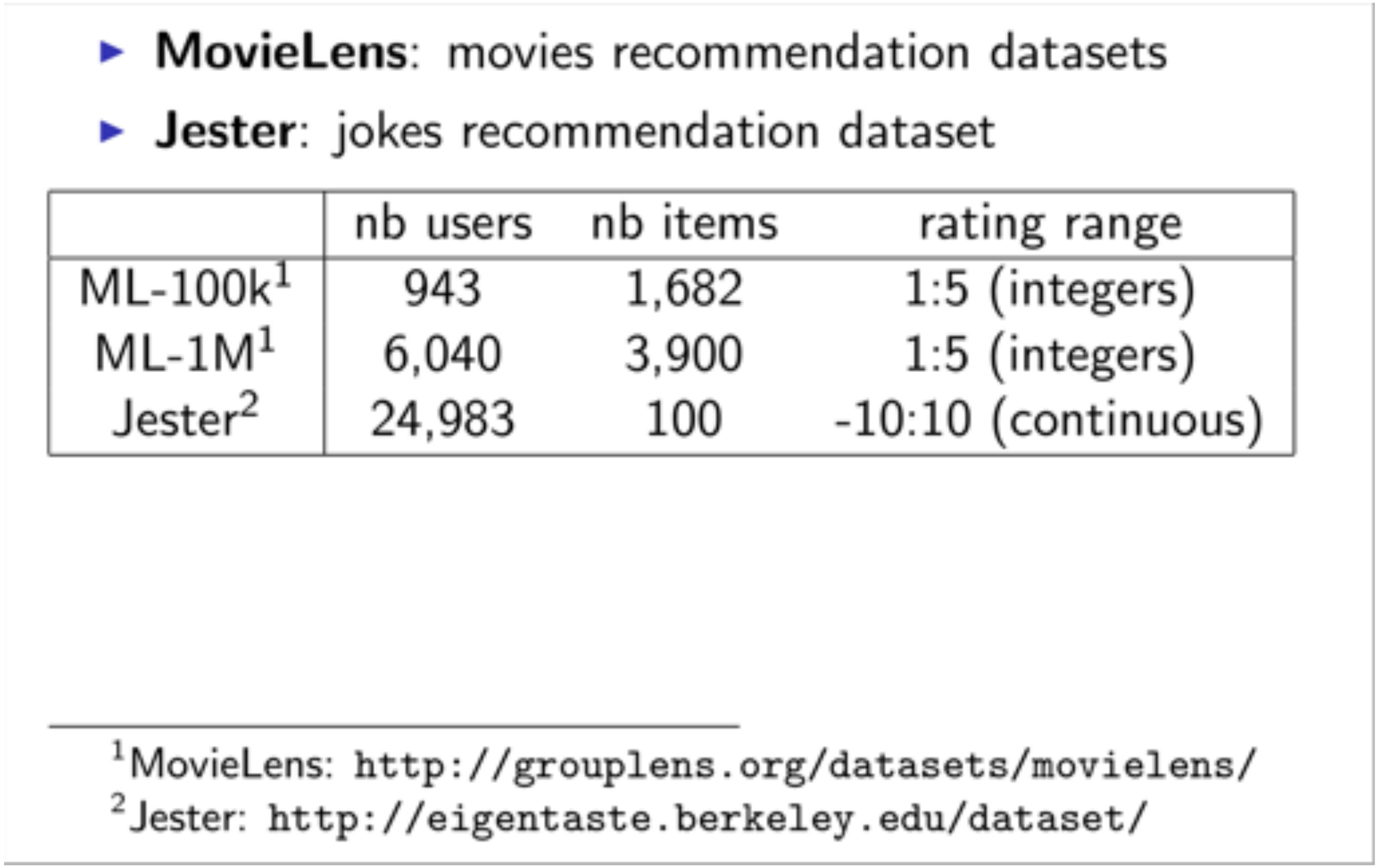
0.6

0.18

B

0.25

# Evaluation



- **MovieLens**: movies recommendation datasets
- **Jester**: jokes recommendation dataset

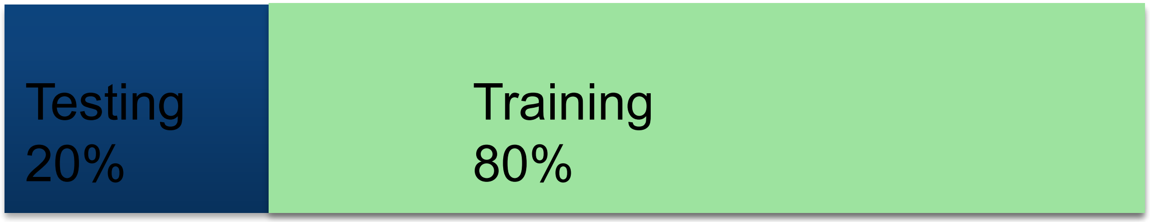
| | nb users | nb items | rating range |
|---|---|---|---|
| ML-100k[1] | 943 | 1,682 | 1:5 (integers) |
| ML-1M[1] | 6,040 | 3,900 | 1:5 (integers) |
| Jester[2] | 24,983 | 100 | -10:10 (continuous) |

[1] MovieLens: `http://grouplens.org/datasets/movielens/`

[2] Jester: `http://eigentaste.berkeley.edu/dataset/`

# Evaluation

1- Split dataset randomly



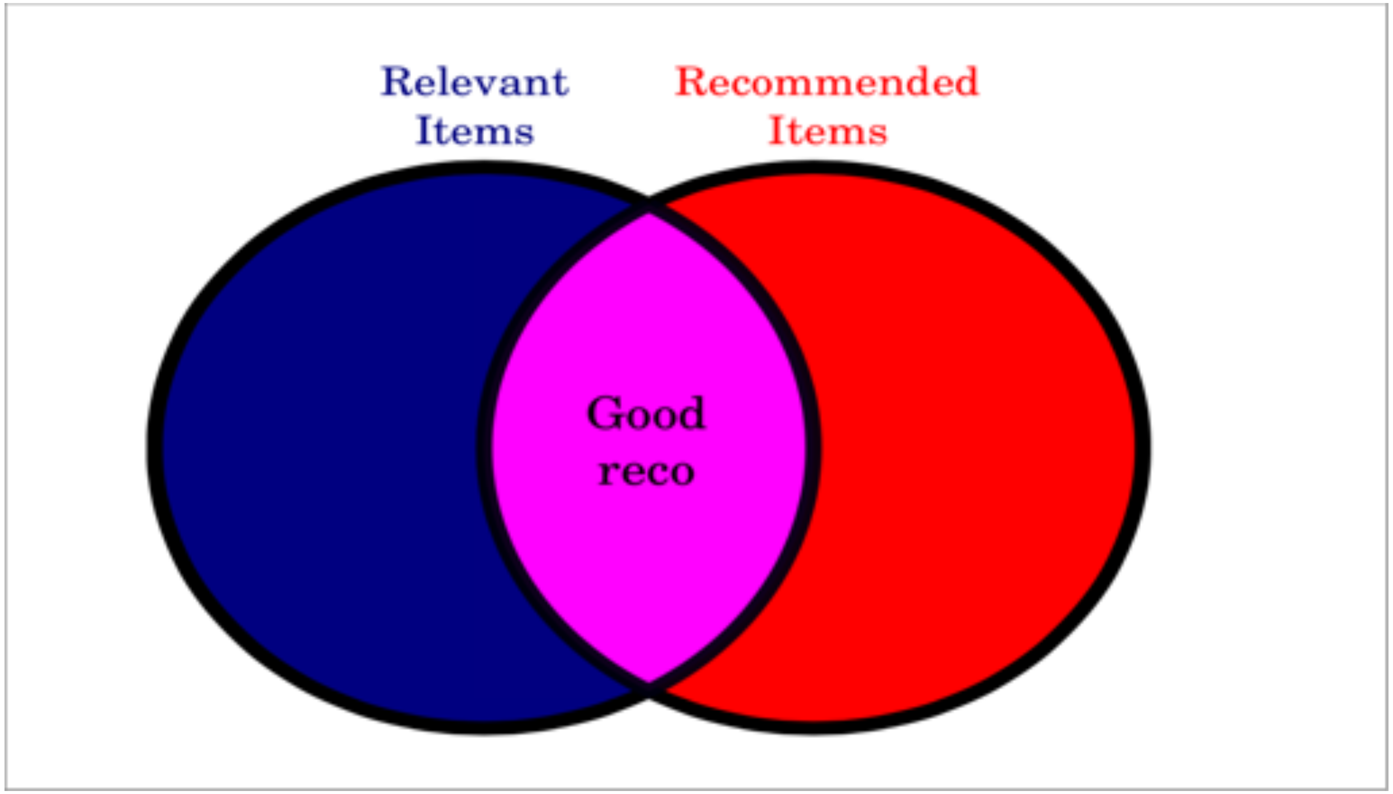
Testing
20%

Training
80%

2- Use training set to fill profiles

3- Generate recommendations and check against training set

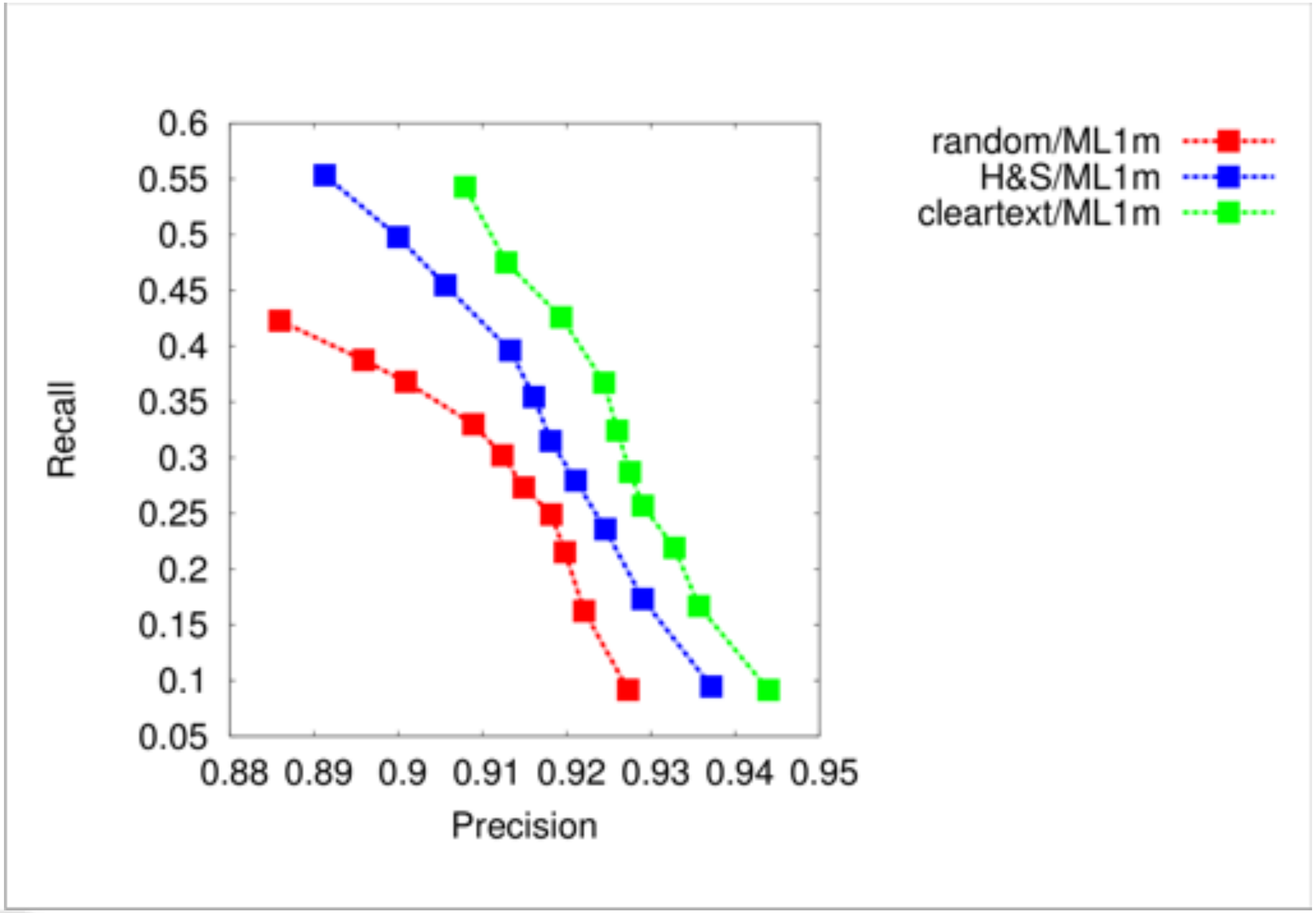
# Metrics



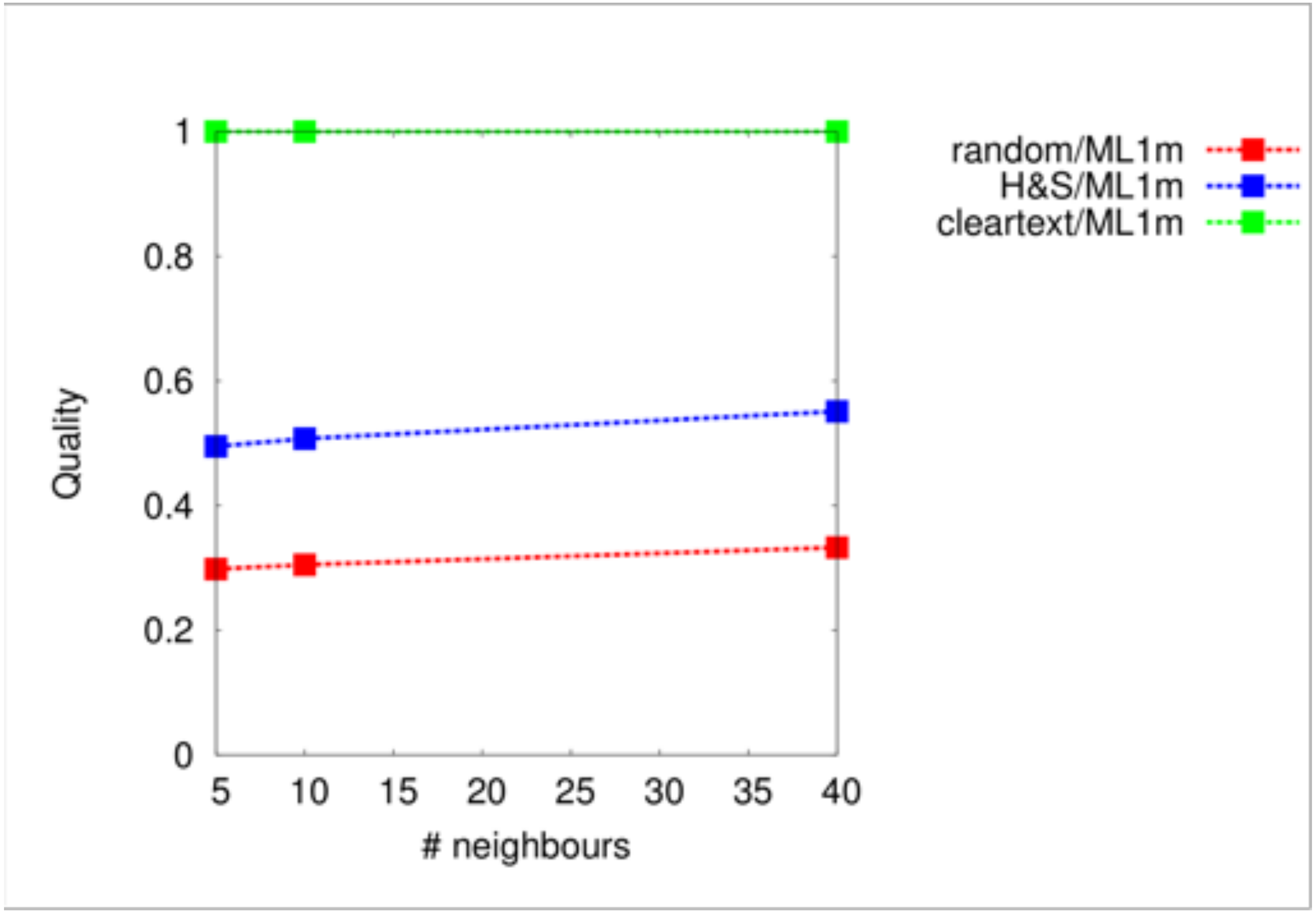
Recall = Good / Relevant

Precision = Good / Recommended

# Recommendation Quality



# Neighborhood Quality



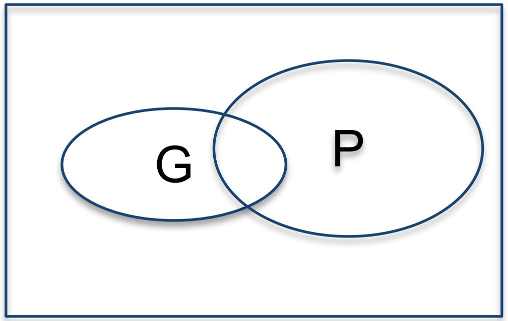
# Privacy: Profile Reconstruction

Profile Reconstruction Attack

- Infer target profile from landmark similarities
- Guess
- items that form the target compact profile
- Assumption: The attacker knows all the item signatures
- Attack:
- Consider closest landmark profile as target profile
- Guess all items that march target profile

# Privacy

- How to measure privacy?
- Simulation: set score
- G = guessed profile
- P = peer profile



$$\textsc{setScore}(G, P) = \frac{|G \Delta P| - |G \cap P|}{|G \cup P|}$$

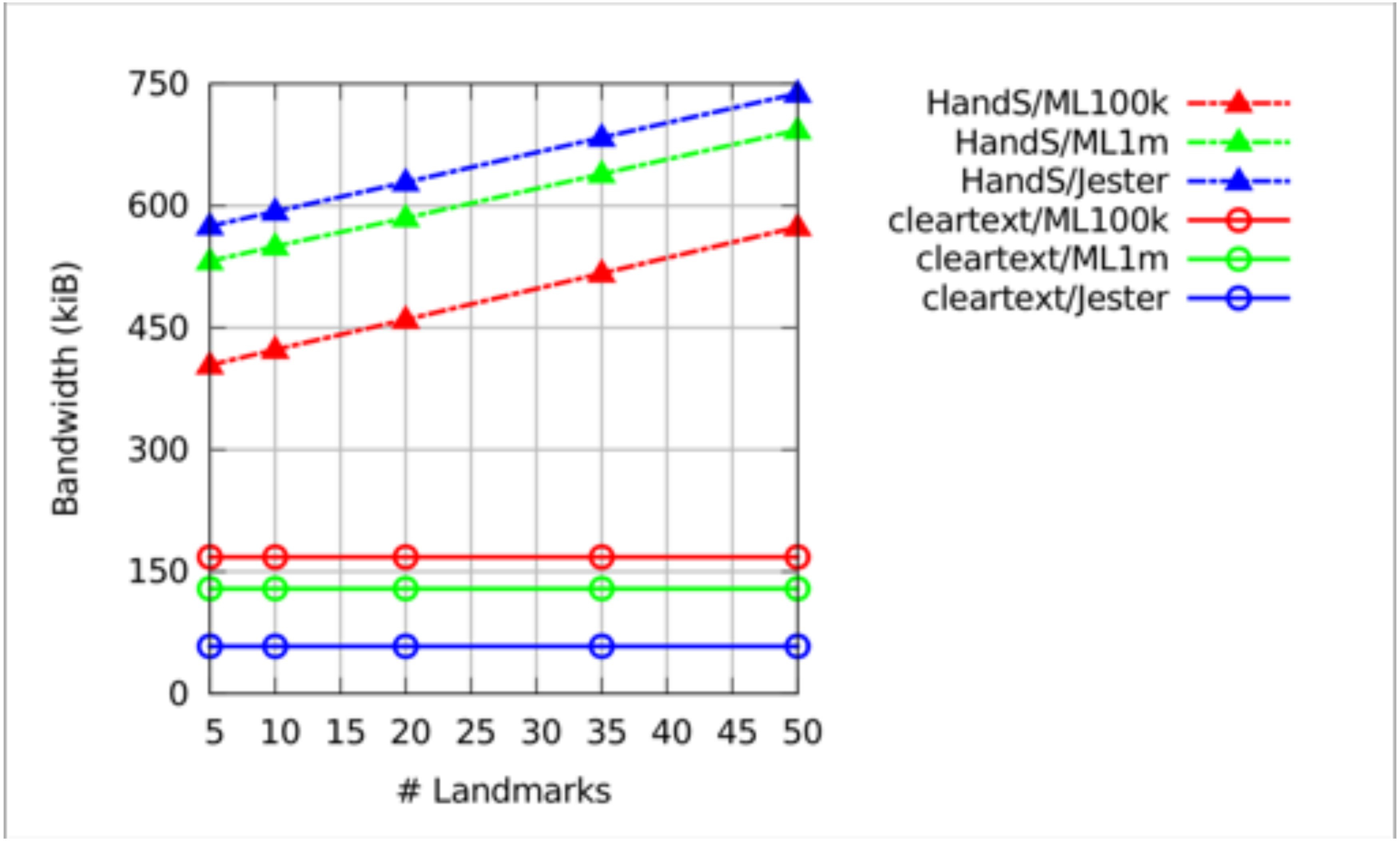
- Range [-1, 1]

# Setup

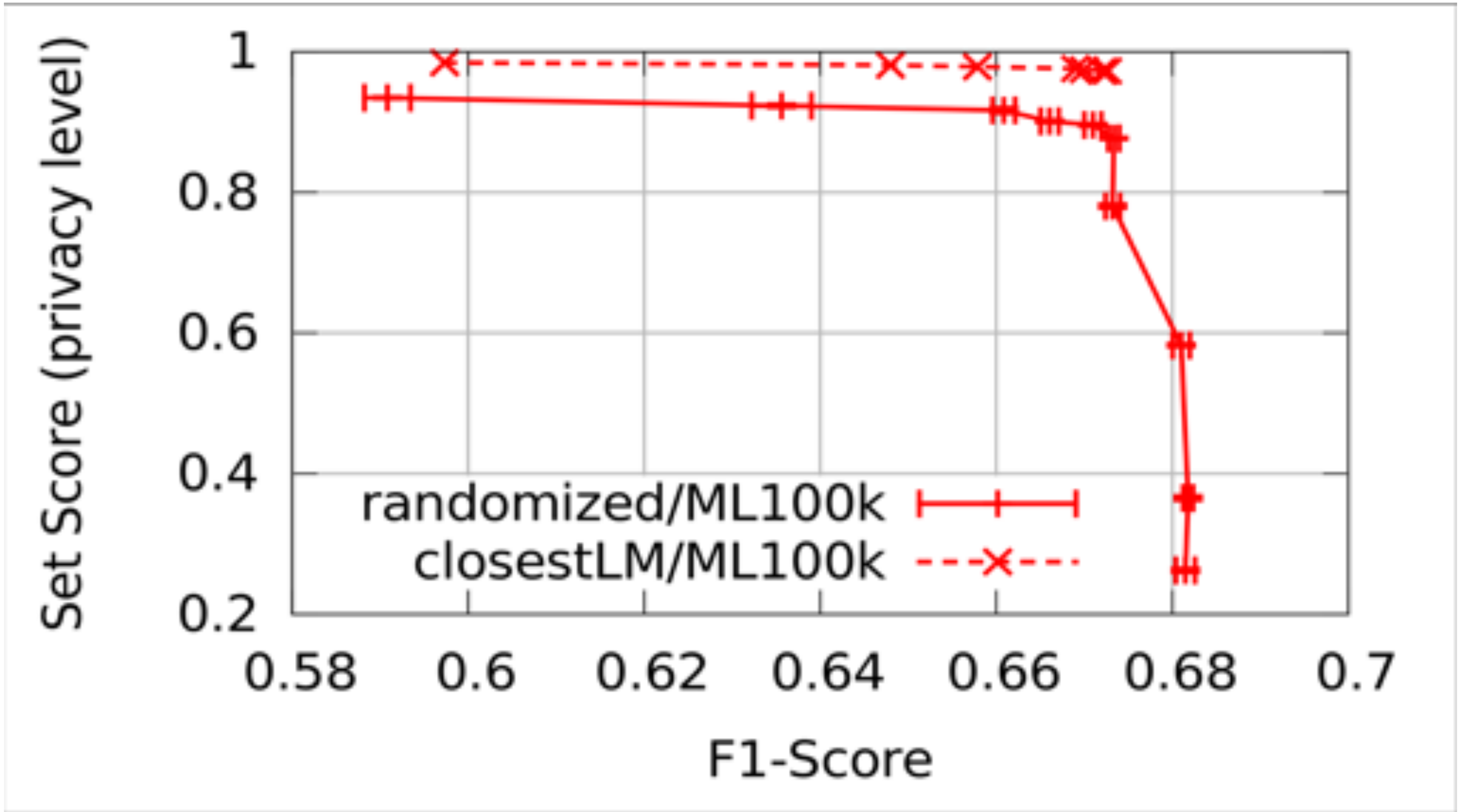
- Baseline: Randomized profiles
- Apply random perturbation to compact profiles
- Varying percentage of randomized bits (5% to 100%)

- Hide and Share configuration
- Vary landmarks between 2 to 100

# Bandwidth Consumption

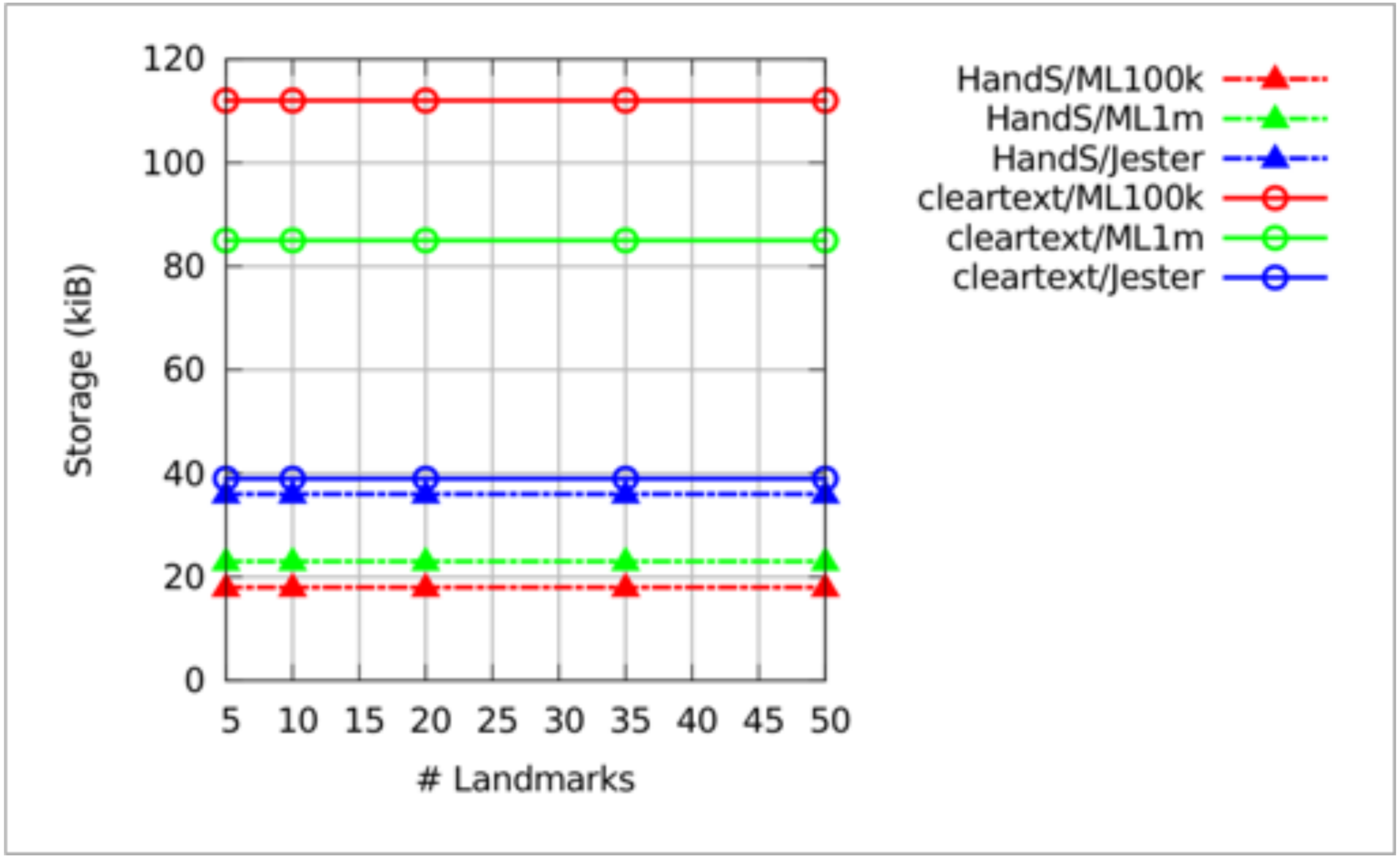


# Results



Set Score (privacy level)
1
0.8
0.6
0.4
0.2
0.58
0.6
0.62
0.64
0.66
0.68
0.7
F1-Score
randomized/ML100k
closestLM/ML100k

# Storage Space


Storage (kiB)
# Landmarks
HandS/ML100k
HandS/ML1m
HandS/Jester
cleartext/ML100k
cleartext/ML1m
cleartext/Jester