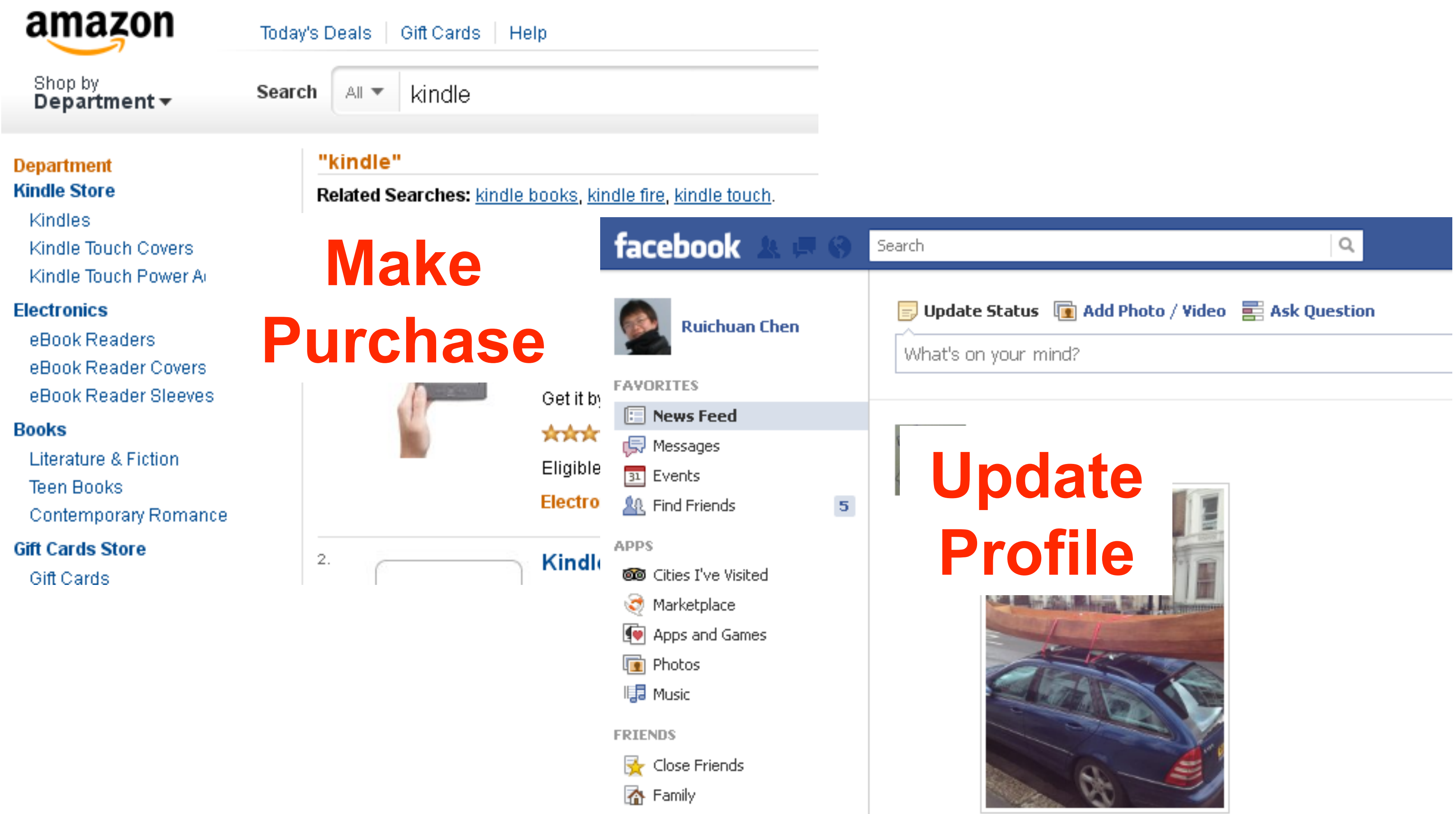
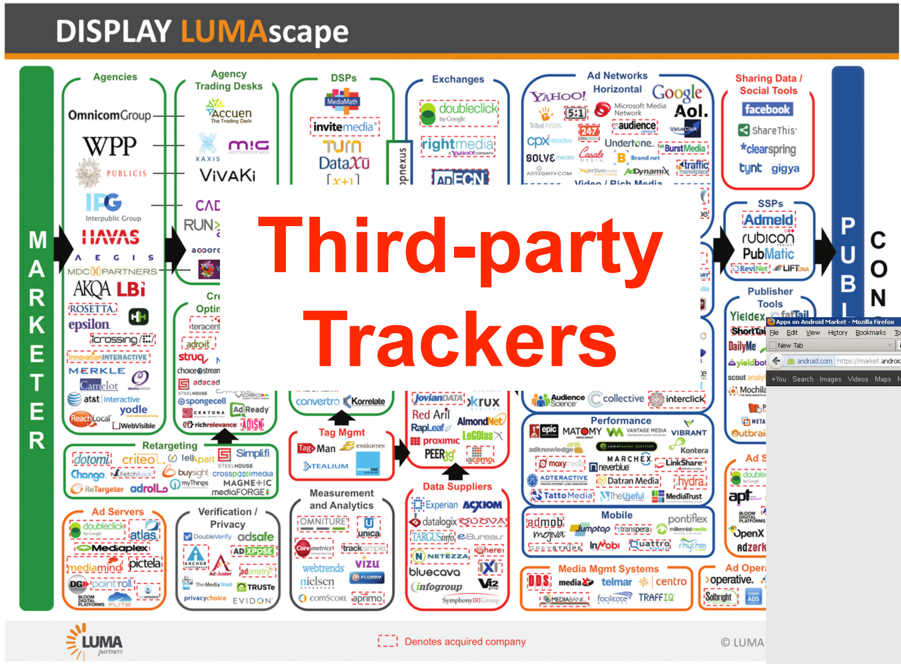
Towards Statistical Queries over Distributed Private User Data

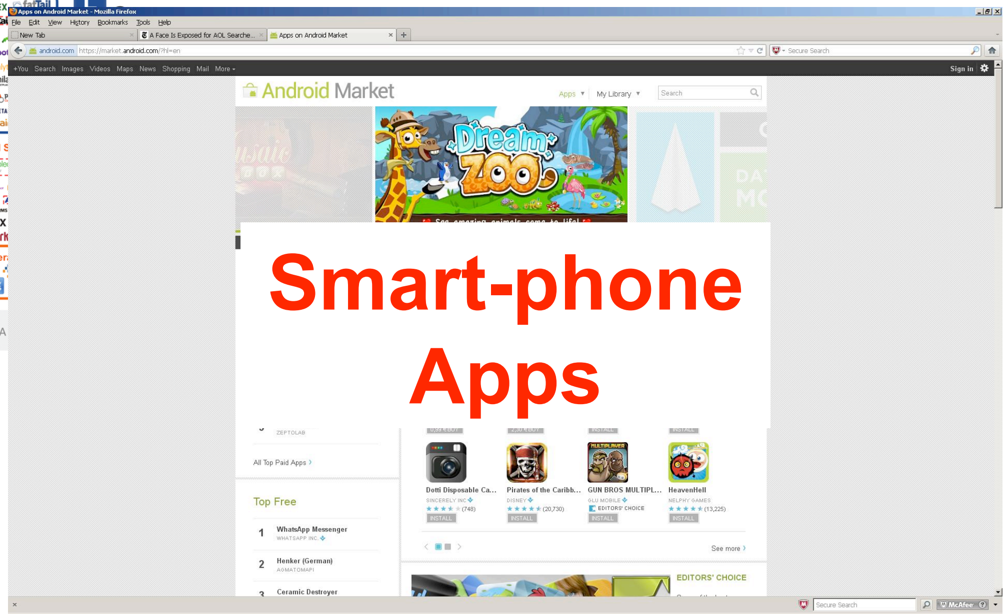
Ruichuan Chen (MPI-SWS)
Alexey Reznichenko (MPI-SWS)
Paul Francis (MPI-SWS)
Johannes Gehrke (Cornell Univ.)

User privacy has become a major concern



Often, users are unaware of data exposure



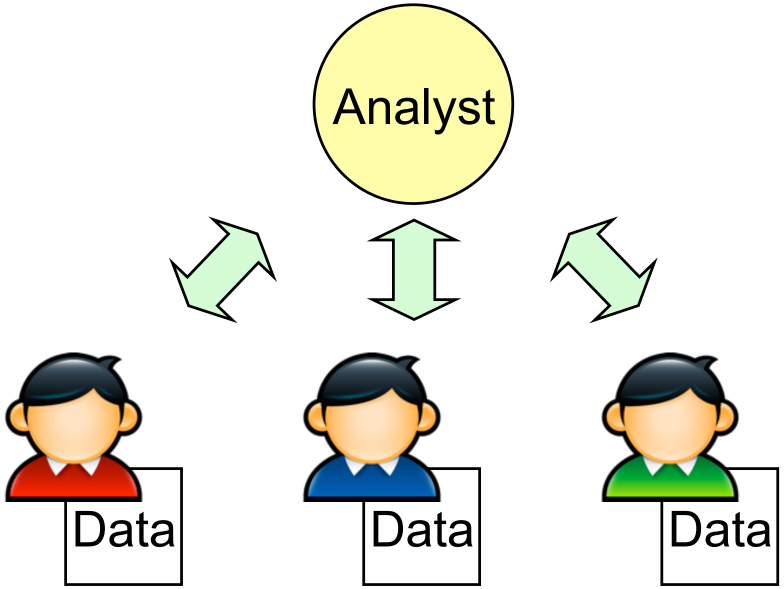


A growing sense

- Privacy loss has to be brought under control!

- User-owned and operated principal
  - Personal data should be stored in a local host (or a cloud device) under the user’s control.

Motivation and problem



- Distributed private user data is important.
- How to make statistical queries over such distributed private user data while still preserving privacy?

Outline

- Related work
- PDDP system
  - Key insights
  - System workflow
  - Implementation, deployment and results
- Conclusion

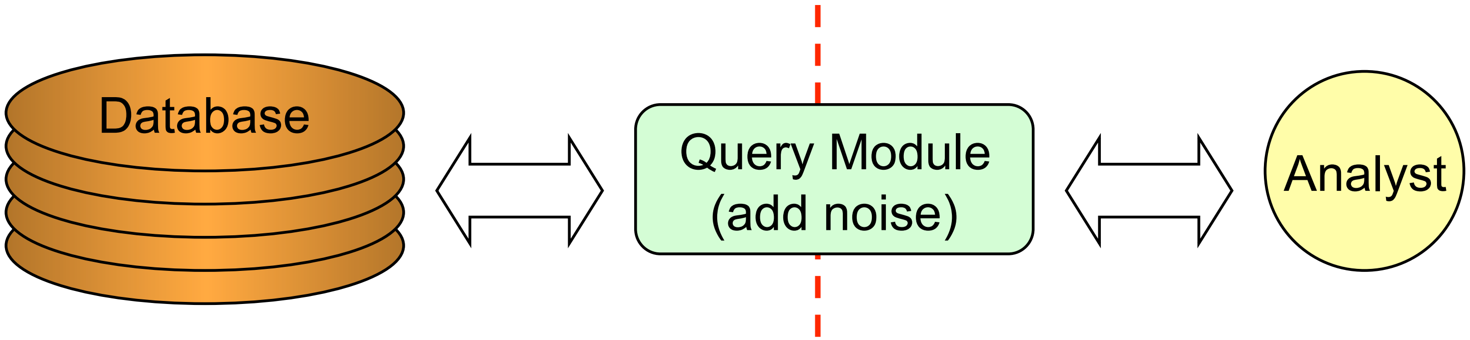
Related work

- Randomization
- K-anonymity, L-diversity, T-closeness

- Differential privacy

Differential privacy

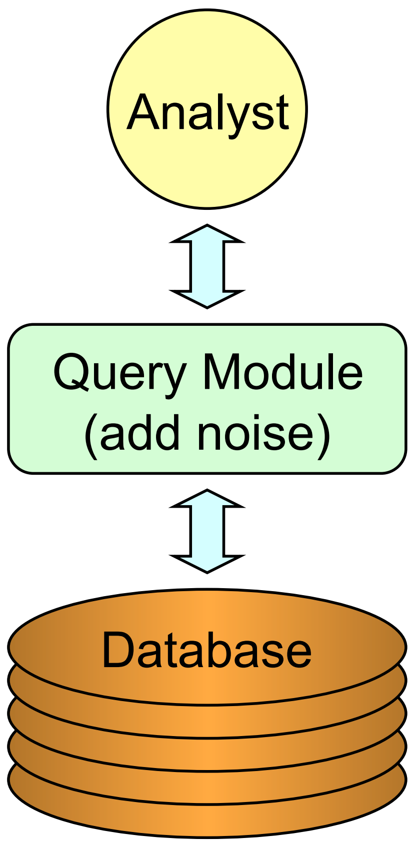
- Differential privacy adds noise to the output of a computation (i.e., query).



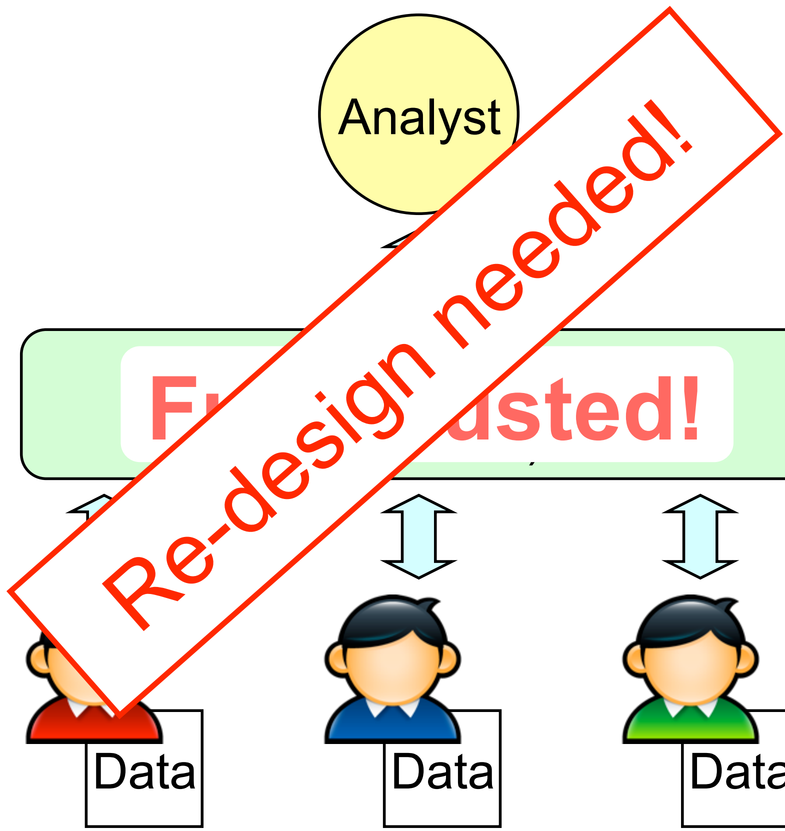
  - Hides the presence or absence of a user.
  - Makes no assumptions about adversary.

Differential privacy in distributed setting

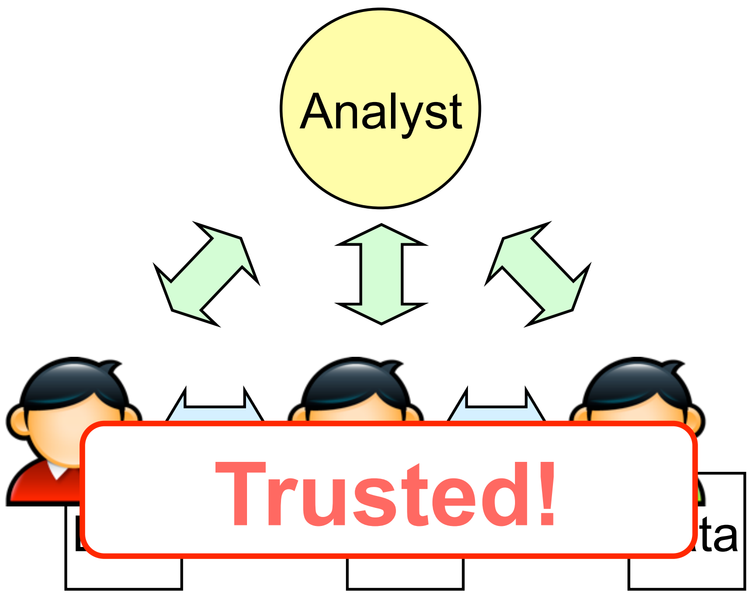
Centralized Environment



Distributed Environment



Prior distributed DP designs



- Scale poorly
  Dwork et al., EUROCRYPT’06.
- Not tolerate churn
  Rastogi and Nath, SIGMOD’10;
  Shi et al., NDSS’11.
- Even a single malicious user can substantially distort the query result
  Rastogi and Nath, SIGMOD’10;
  Shi et al., NDSS’11;
  Götz and Nath, MSR-TR’11.

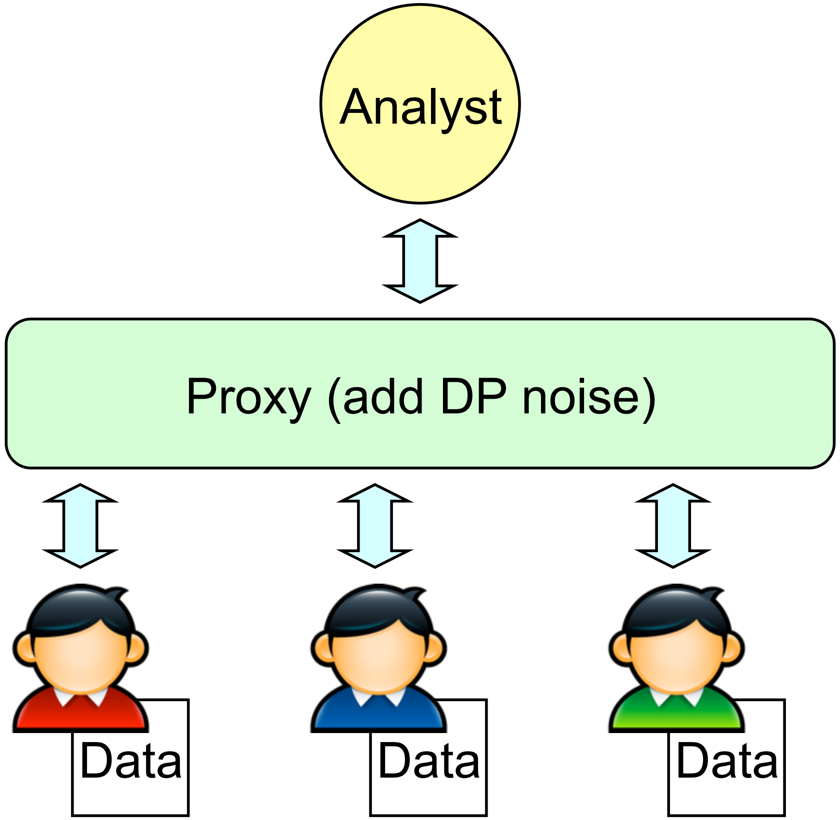
Outline

- Related work
- PDDP system
  - Key insights
  - System workflow
  - Implementation, deployment and results
- Conclusion

PDDP system

- PDDP: Practical Distributed Differential Privacy
  - Operates at large scale
  - Tolerates churn
  - Puts tight bound on the extent to which a malicious user can distort query results

Components & assumptions



Analyst is potentially malicious
(violating user privacy)

Proxy is honest but curious
1) Follows the specified protocol
2) Tries to exploit additional info
that can be learned in so doing

Clients are user devices.
Clients are potentially malicious
(distorting the final results)

Outline

- Related work
- PDDP system
  - Key insights
  - System workflow
  - Implementation, deployment and results
- Conclusion

Key insights – binary answer

- How to limit query result distortion?

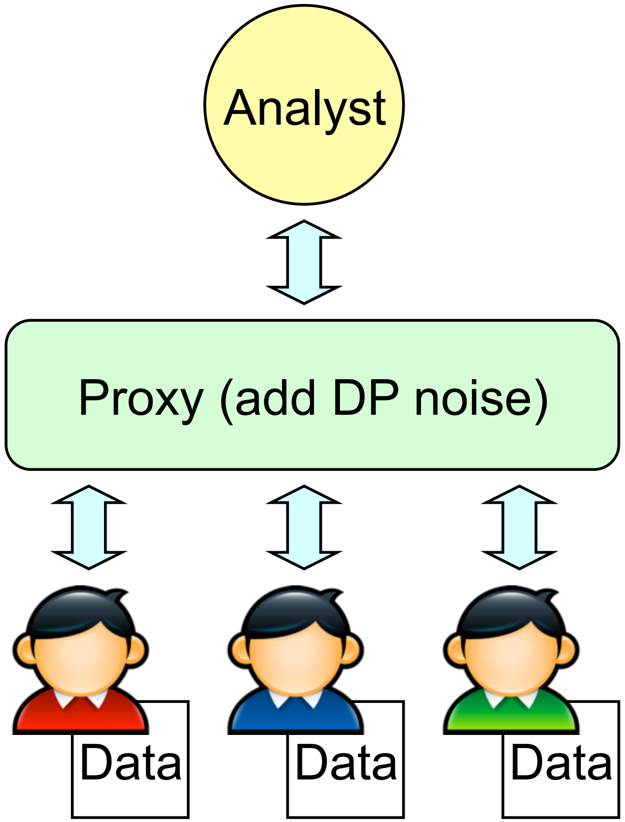
- Solution:
  - Ensure that a client cannot arbitrarily manipulate answers.
  - Split answer's value range into buckets.
  - Enforce a binary answer in each bucket.
    - Zero-knowledge proofs X
    - Bit-cryptosystem √

Key insights – binary answer

- Query: “how old are you?”
  - 4 buckets: 0~12, 13~20, 21~59, and ≥60.
  - Answers: a ‘1’ or ‘0’ per bucket.
    - 30 years-old → 0, 0, 1, 0
  - Malicious clients cannot substantially distort the query result!

Key insights – blind noise

- How to achieve differential privacy?



- Solution:
  - An anonymizing honest-but-curious proxy.
  - Proxy generates additional binary answers in each bucket as differentially private noise.

What if analyst publishes noisy result?

Proxy knows noise

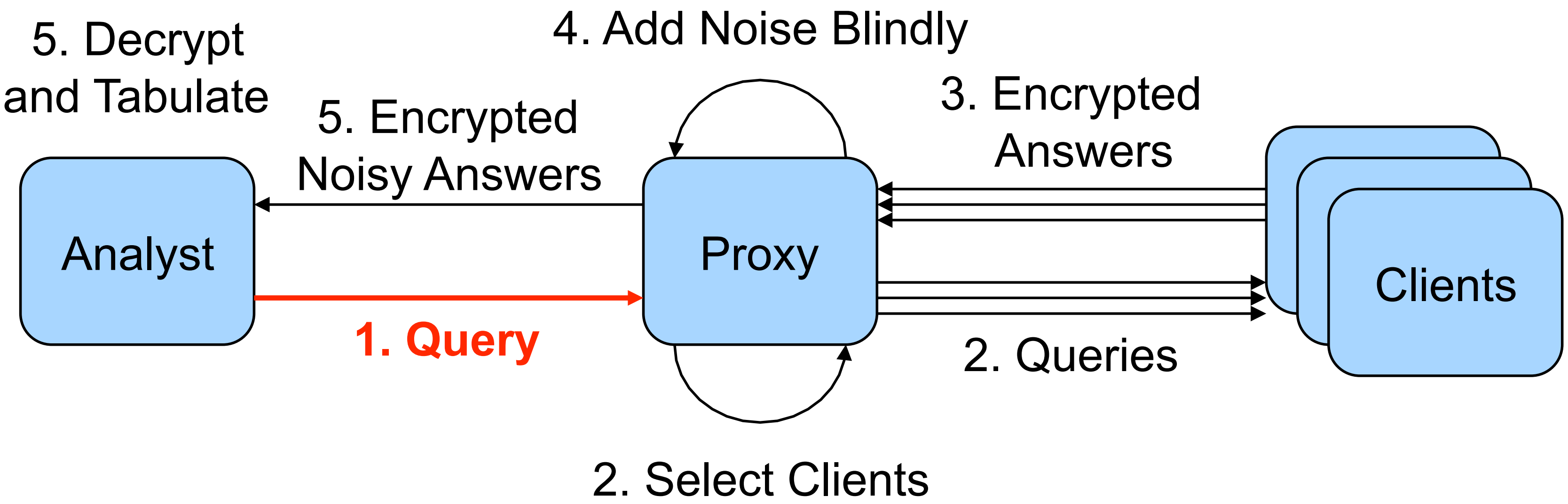
Proxy knows noisy-free result

Blind noise addition!

Outline

- Related work
- PDDP system
  - Key insights
  - System workflow
  - Implementation, deployment and results
- Conclusion

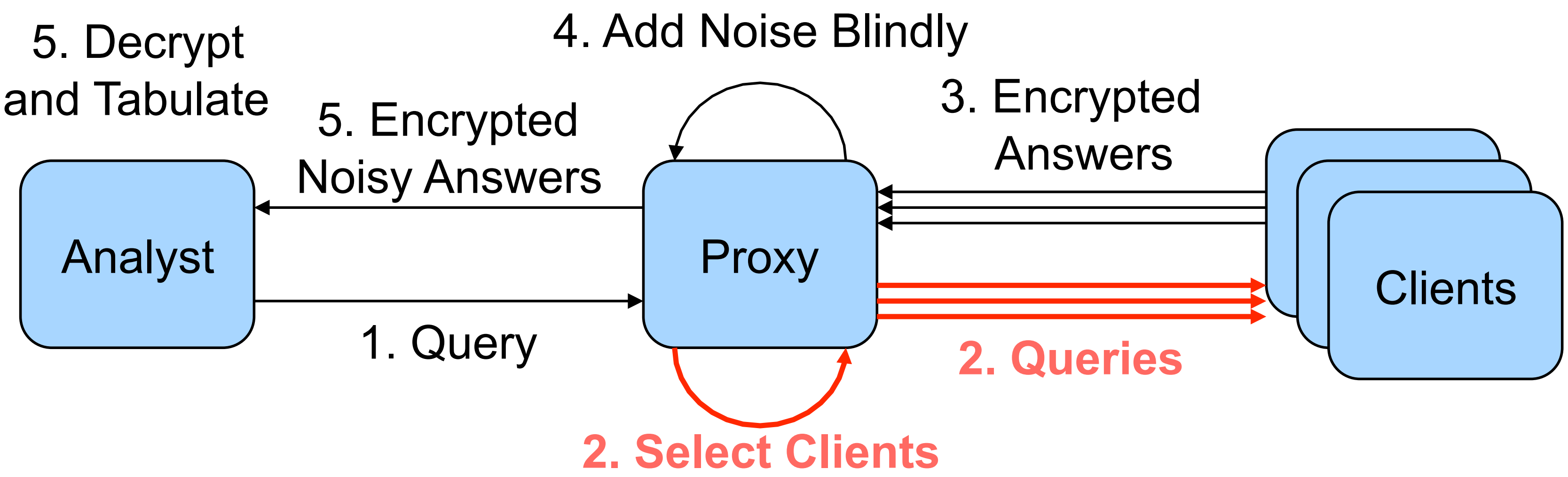
Step 1: query initialization



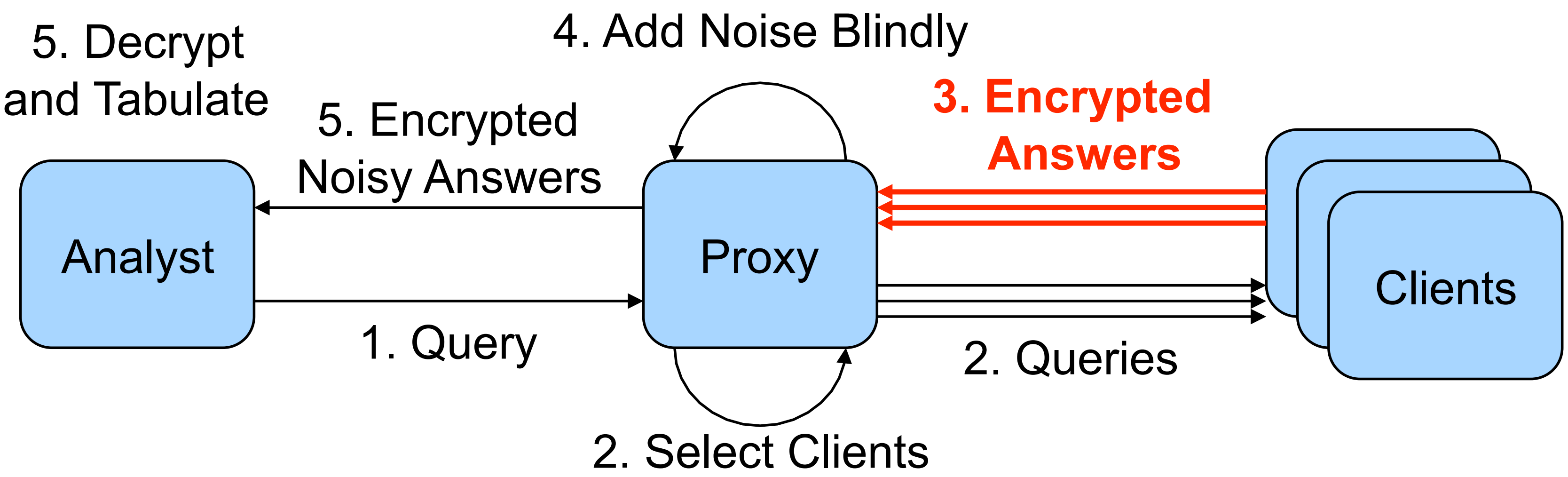
Step 1: query initialization (cont.)

- Example: age distribution among males?
  - Query: SELECT age FROM local_db WHERE gender='m'
  - Buckets: 0~12, 13~20, 21~59, and ≥60
  - # clients queried (c): 1000
  - DP parameter (ε): 1.0

Step 2: query forwarding



Step 3: client response



Step 3: client response (cont.)

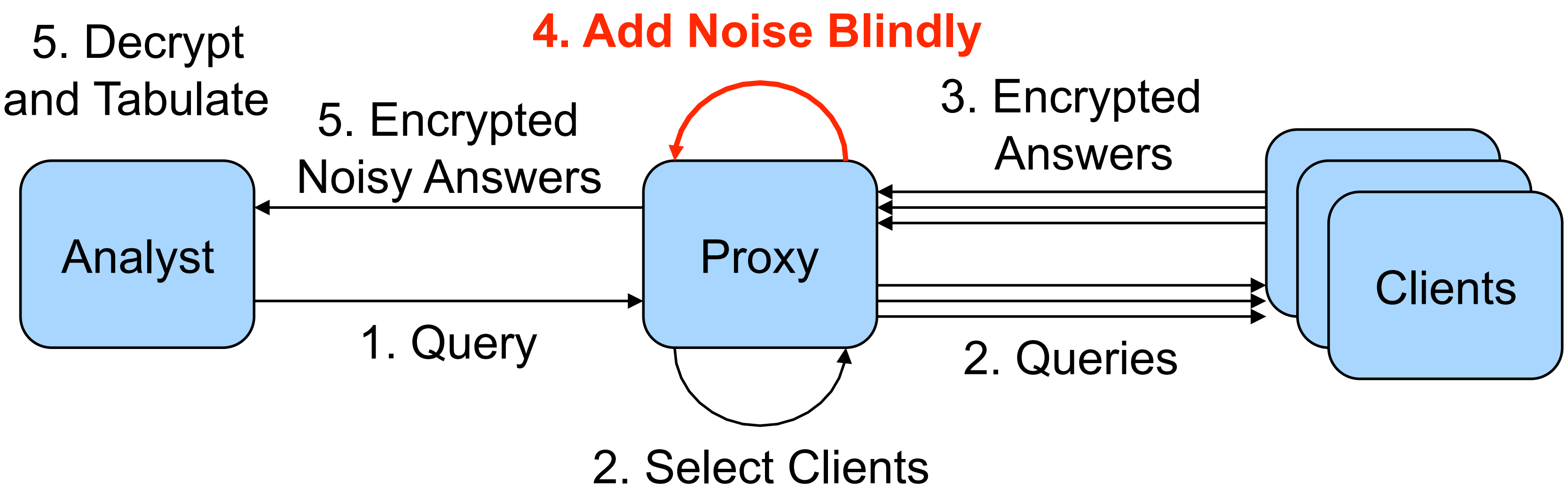
- Client executes query over its local data and produces answer
  - A '1' or '0' per bucket
  - More than one bucket may contain a '1'

Step 3: client response (cont.)

- Per-bucket answer value is individually encrypted with the analyst’s public key.

- Goldwasser-Micali (GM) cryptosystem
  [Goldwasser and Micali, STOC’82]
  - Single-bit cryptosystem
    - Enforce a binary answer in each bucket
  - Very efficient
  - XOR-homomorphic
    - $E(a) * E(b) = E(a \oplus b)$

Step 4: blind noise addition



Step 4: blind noise addition

- Proxy adds DP noise to each bucket.
  - Generate some additional binary answers (i.e., '0' or '1') as DP noise, called coins.
    - Coins must be unbiased.
    - Coins are encrypted with analyst's public key.
  - How many coins needed?

$$n = \lfloor \frac{64 \ln(2c)}{\varepsilon^2} \rfloor + 1$$

c: # clients queried
ε: DP parameter

- Question: how to generate coins blindly?

Coin generation

- Straightforward approaches
  - Proxy generates coins?
    - Curious proxy could know noise-free result!
  - Clients generate coins?
    - Malicious clients could generate biased coins!

Collaborative coin generation

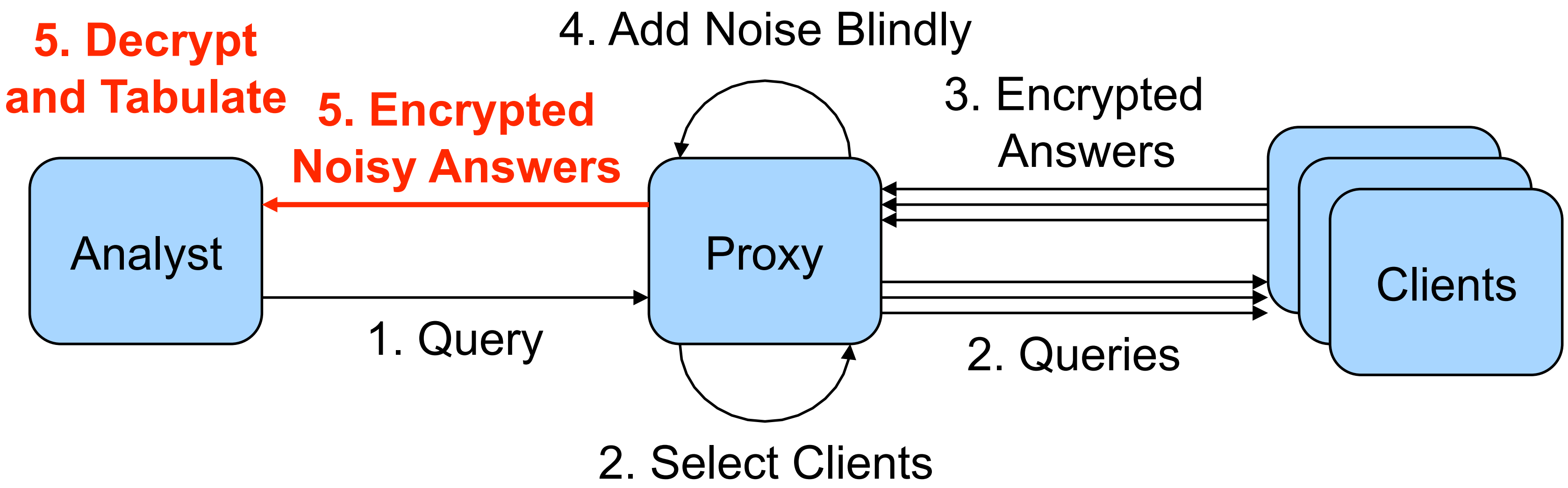
- Our approach
  - Each online client periodically generates an encrypted unbiased coin $E(o_c)$
  - Proxy blindly re-flips the coin $E(o_c)$
    - Generate an unbiased coin $E(o_p)$ locally
    - Multiply $E(o_c)$ with $E(o_p)$
    - The product $E(o_c) * E(o_p)$ is an unbiased coin

Collaborative coin generation

- GM cryptosystem is XOR-homomorphic
  - $\underbrace{E(o_c)}_{\text{Possibly biased}} * \underbrace{E(o_p)}_{\text{Unbiased}} = \underbrace{E(o_c \oplus o_p)}_{\text{Unbiased}}$

- Proxy doesn't know the actual value of the generated unbiased coin
  - Curious proxy cannot know noise-free result

Step 5: noisy answers to analyst



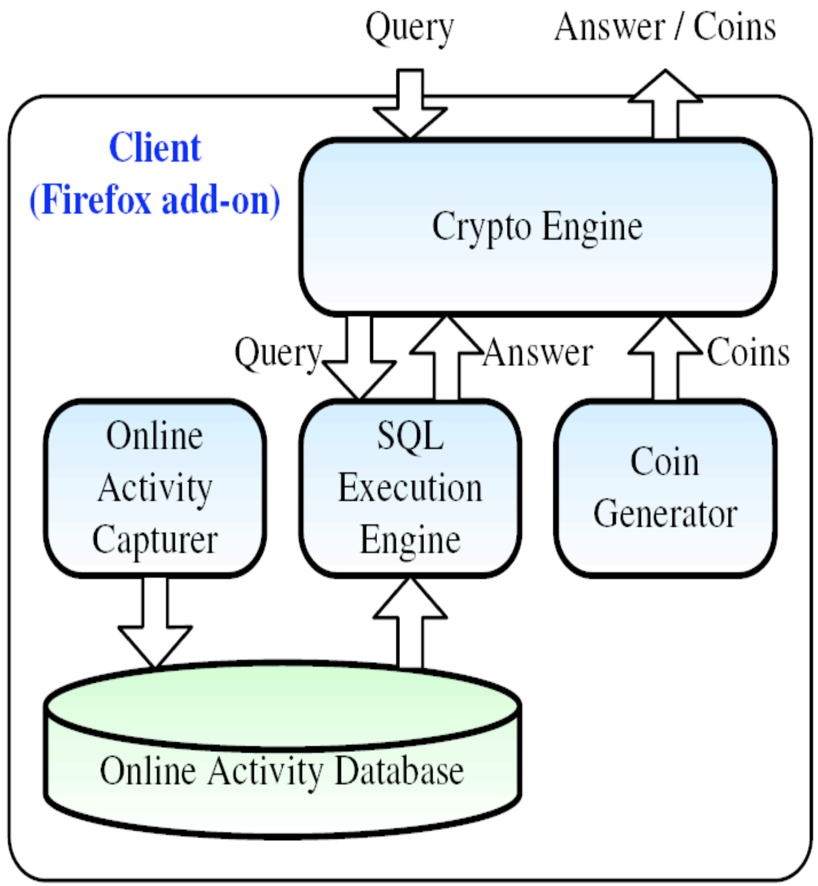
- Each bucket: client answers + coins (noise)
- In the end, analyst obtains the noisy answer for how many clients fall within each bucket.

Outline

- Related work
- PDDP system
  - Key insights
  - System workflow
  - Implementation, deployment and results
- Conclusion

Implementation & deployment

- Client
  - Firefox add-on (9.6K LOC)
  - SQLite storage



Available at http://www.mpi-sws.org/~rchen/pddp/pddpFX.xpi

Implementation & deployment

- Proxy
  - Web service on Tomcat (3.6K LOC)
  - Proxy state in MySQL database
- Analyst
  - Java program (800 LOC)
- Deployment
  - 600+ real clients

Client performance

- Major concern: crypto operations
- Performance at client

Firefox	Chrome	Smart phone
2157.96	22773.86	808.87

encryptions / second

Proxy/analyst performance

Encryption	Decryption	Homomorphic Op
15323.32	6601.10	123609.39

operations / second

- Example:
  - 1M clients, 10 buckets, and $\varepsilon = 1.0$
  - Computation: < 30 CPU-minutes
  - Bandwidth and storage: 1.2GB

Query exercise

- 5 queries towards client deployment
  - Many low-activity clients
    - 30% of clients visited ≤10 webpages
  - Many clients visited just a few websites
    - 47% of clients visited ≤10 websites
  - Most browsing on a user's top 3 favorite websites
  - Search engine is often used
  - Google ads are shown relatively often

Outline

- Related work
- PDDP system
  - Key insights
  - System workflow
  - Implementation, deployment and results
- Conclusion

Conclusion

- PDDP: the first practical distributed differentially private (query) system
  - Scales well
  - Tolerates churn
  - Places tight bound on malicious user’s capability

- Key insights:
  - Binary answer in bucket
  - Blind noise addition