Privacy-Preserving Database Fingerprinting

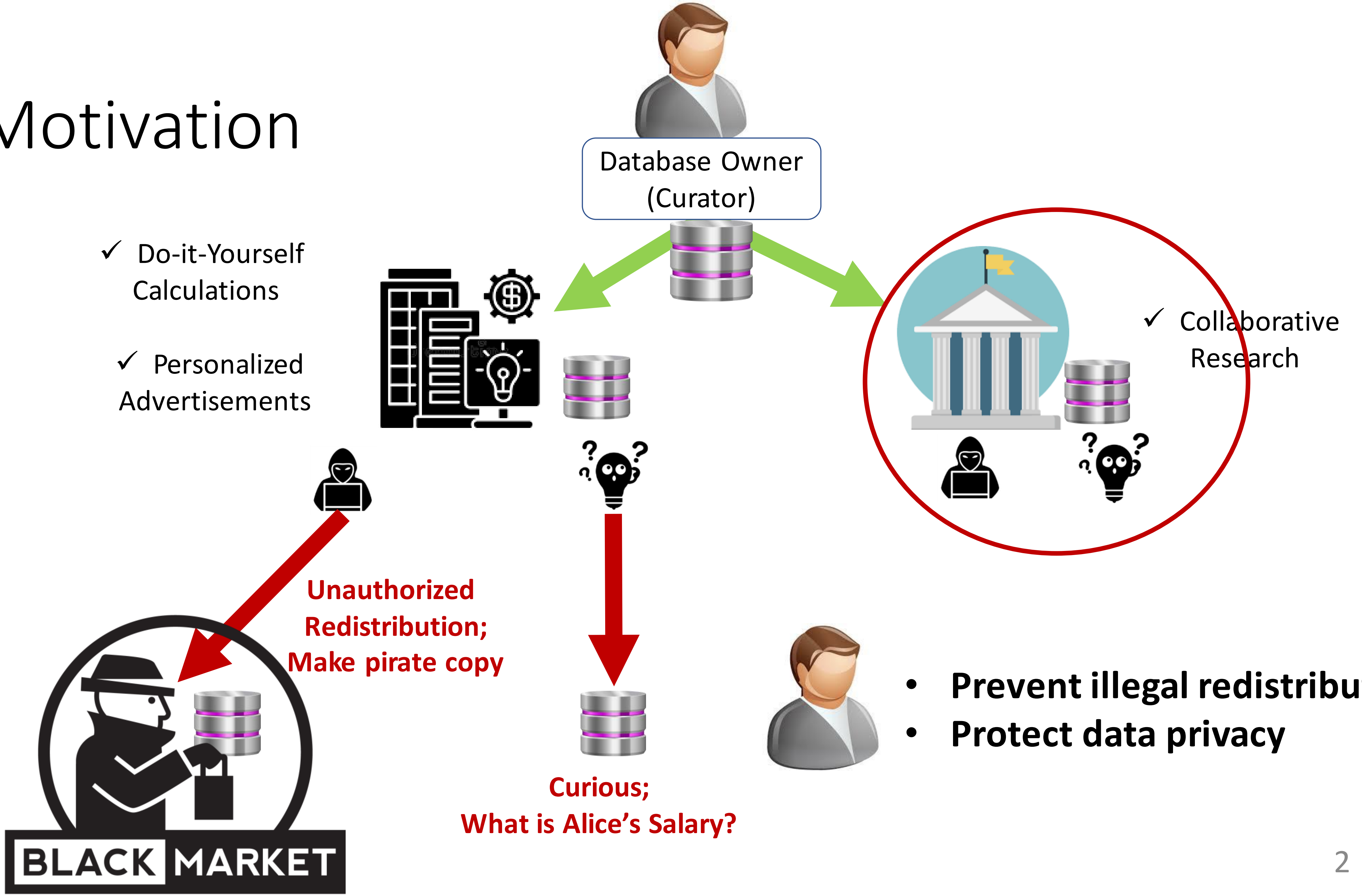
Tianxi Ji[1], Erman Ayday[2], Emre Yilmaz[3], Ming Li[4], Pan Li[2]

[1]Texas Tech University, [2]Case Western Reserve University,

[3]University of Houston-Downtown, [4]University of Texas at Arlington

NDSS 2023, San Diego, CA, USA

Motivation



- **Prevent illegal redistribution**
- **Protect data privacy**

Techniques

- Database Fingerprinting
  - Imperceptible
  - Prevent illegal redistribution
  - Identify source of data leakage
  - Hold the traitor(s) liable for redistribution

- Differential Privacy (DP)
  - Obfuscate individuals’ data
  - Defend against adversarial inference

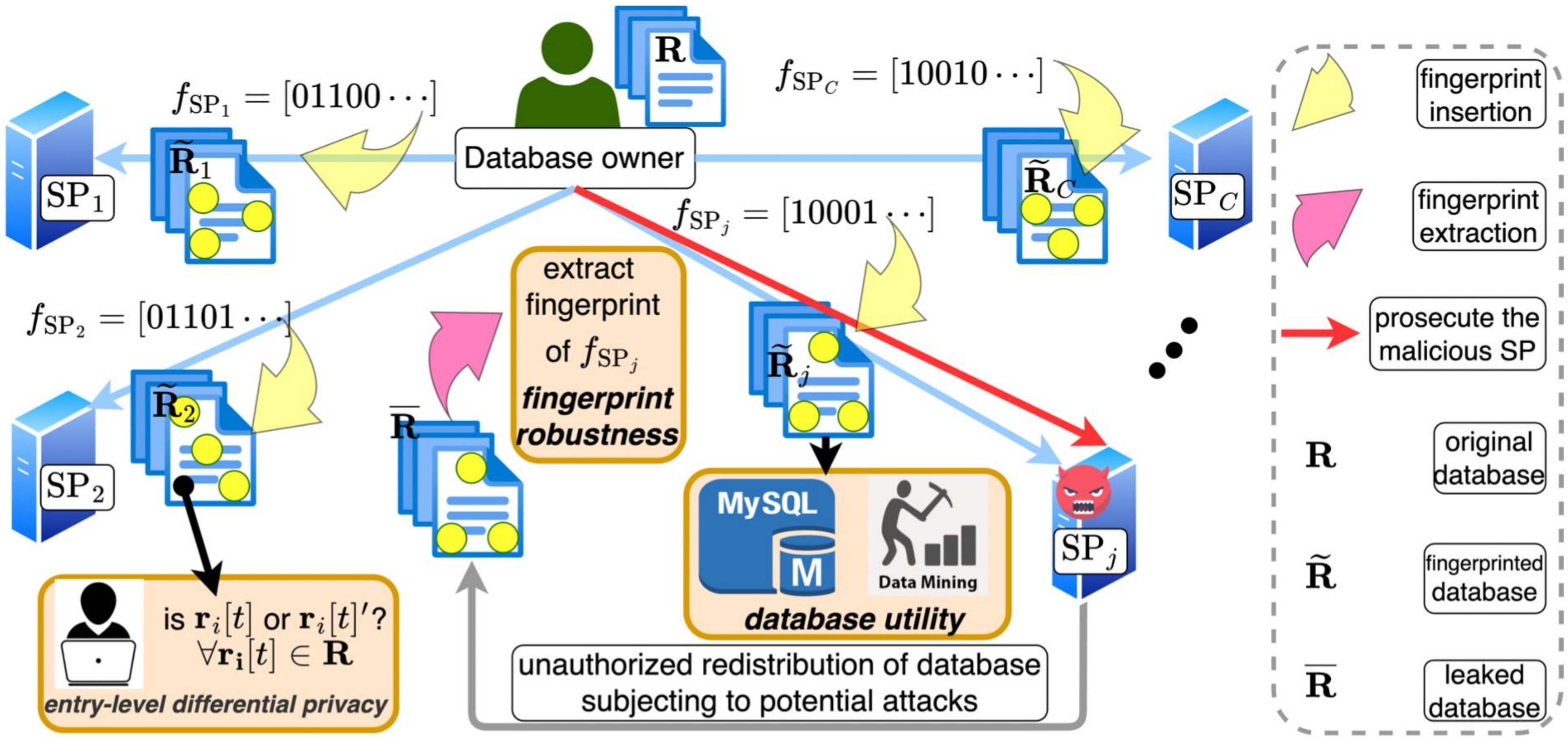
Challenges

- **Prevent illegal redistribution**
- **Protect data privacy**

- Orthogonal objectives
  - Liability via fingerprinting requires adding different noises to all copies
    i.e., recipients receive **different** copies of DBs
  - Privacy via data sanitization requires adding noise once
    i.e., recipients can receive the **same** copy of DB
- Both fingerprinting and DP compromise DB utility
  - Sequential approach (fingerprinting followed by DP) is suboptimal
- Need a **unified** scheme to maintain DB utility

Privacy-Preserving DB Fingerprinting

Privacy-Preserving DB Fingerprinting



Definitions

- Relational DB
  - A collection of T-tuples, each is an individual
  - Each record has an **immutable** pseudo-id, i.e., **primary key**
- Neighboring relational DB
  - Two DBs differ only by one entry (an attribute of a single individual)
- Sensitivity of relational DB
  - The maximum change of an entry
- ϵ-entry-level DP: $\Pr[\mathcal{M}(\mathbf{R}) = S] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathbf{R}') = S]$

$$\mathbf{R}, \mathbf{R}' \in \mathcal{D}, \mathcal{S} \in \mathrm{Range}(\mathcal{M}), \epsilon > 0$$

Intermediate scheme: bit-level randomization

- Design principle
  - Fingerprinting schemes performs XOR between insignificant bits of data w. binary marks
  - Random: selection of bits and value marks
  - The randomness can be leverage to achieve privacy
- A bit-level randomization scheme pseudorandomly selects some bits of data entries and changes their values by XORing them with random binary marks, B, and $B \sim \mathrm{Bernoulli}(p)$

Theorem: Given R with Δ, bit-level randomization preserves ϵ-entry-level DP if it marks last $K = \lfloor \log_2 \Delta \rfloor + 1$ bits, $p = \frac{1}{e^{\epsilon/K} + 1}$

ϵ-entry-level DP fingerprinting

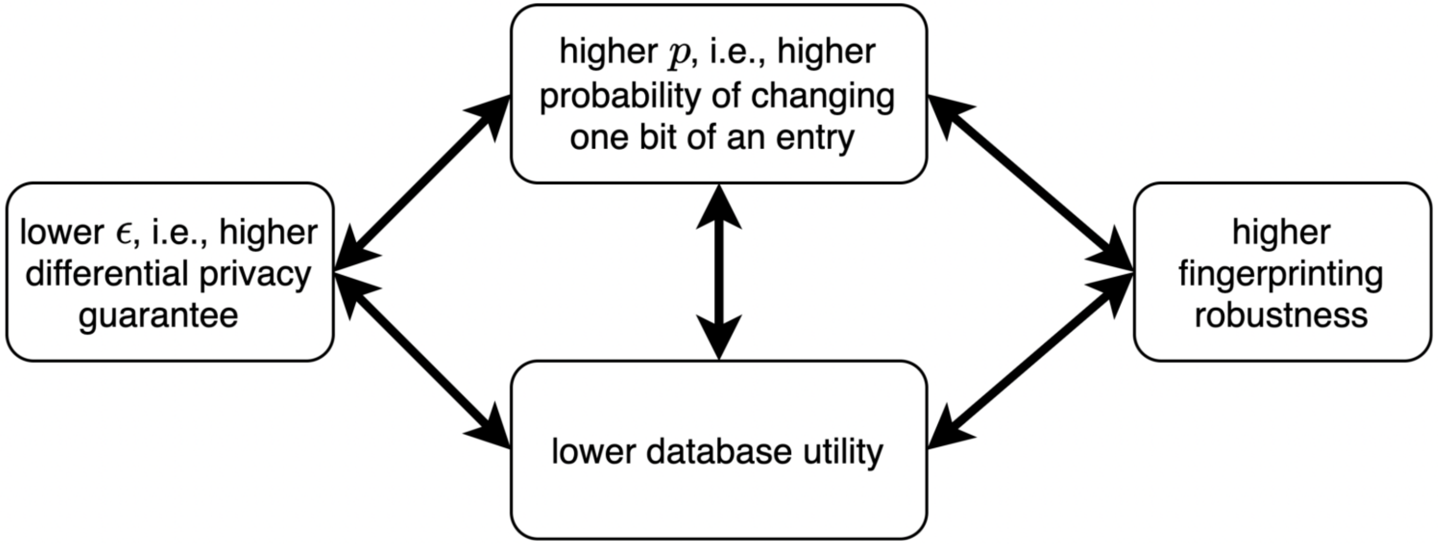
- Collect all fingerprintable bits

$$\mathcal{P} = \left\{ \mathbf{r}_i[t,k] \middle| i \in [1,N], t \in [1,T], k \in [1, \min\{K, K_t\}] \right\}$$

N: # of rows, K_t: # of bits to represent attribute t, $K = \lfloor \log_2 \Delta \rfloor + 1$

- Key steps
  - Generate the fingerprint (binary bit-string) of a SP using Hash function
  - Fingerprint a bit in $\mathcal{P}$ (i.e., $\mathbf{r}_i[t,k] \oplus B$) if a specific condition holds
    - ➢The condition is carefully designed such that $\Pr[B = 1] = \frac{1}{e^{\epsilon/K}+1}$

Theoretical analysis: associating privacy, fingerprint robustness, DB utility



Closed form association between **privacy** (ϵ), **randomization** (p), **robustness** (against random flipping, subset, correlation attacks), and DB **utility** (accuracy, statistics, e.g., marginal/joint distribution)

Cumulative privacy loss due to multiple sharing

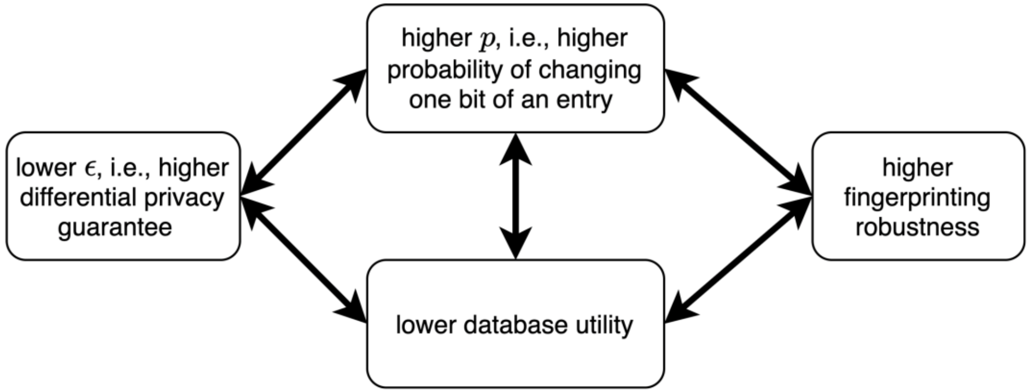
- Practical concern of DP
  - Privacy degrades linearly if the same statistics are repeatedly shared
  - The same is true for repeatedly sharing a DB with multiple SPs

- Resort to Sparse Vector Technique (SVT)
  - Only releases a noisy result when it is beyond a noisy threshold
  - Pays the cost of privacy only for queries satisfying a certain condition, i.e.,

$$function(\mathrm{DB}) + noise_1 \geq \Gamma + noise_2$$

Cumulative privacy loss control via SVT

- Design principle
  - For C SPs asking for the DB
  - Only share fingerprinted copies with certain **privacy** and **robustness** requirements
  - Requirements on **privacy** and **robustness** can be quantified via DB utility



Consider $function(\mathrm{DB}) = \|\mathcal{M}(\mathbf{R}) - \mathbf{R}\|_{1,1}$

Associate with **privacy** (ϵ), **randomization** (p), and **robustness**

Share fingerprinted DB with C SPs via SVT

- Key steps:
  - Generate a fingerprinted copy, $\mathcal{M}(\mathbf{R})$, with privacy budget ϵ
  - Sample two Laplace noises $\mu \sim \mathrm{Lap}(\frac{\Delta}{\epsilon_2})$ and $\rho \sim \mathrm{Lap}(\frac{\Delta}{\epsilon_3})$
  - Only share $\mathcal{M}(\mathbf{R})$ if $\left|\left|\mathcal{M}(\mathbf{R}) - \mathbf{R}\right|\right|_{1,1} + \mu \geq \Gamma + \rho$

Theorem: Preserve is (ϵ_0, δ_0)-entry-level DP.

$$\begin{aligned}\epsilon_0 &= \sqrt{2C\ln(1/\delta')}(\epsilon + \epsilon_2 + \epsilon_3) \\ &\quad + C\Big(\epsilon(e^{\epsilon} - 1) + (\epsilon_2 + \epsilon_3)(e^{\epsilon_2 + \epsilon_3} - 1)\Big) \\ \delta_0 &= 2\delta'\end{aligned}$$

Experiments

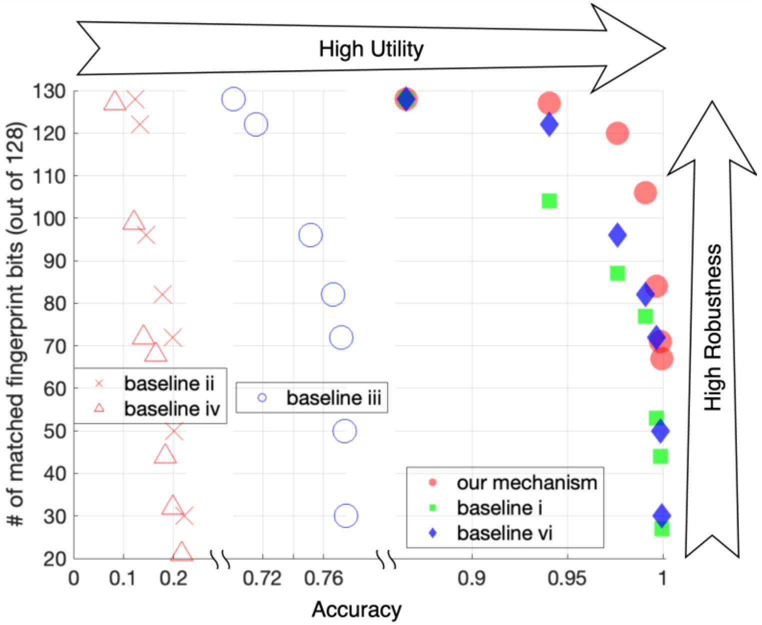
- Two DBs
  - Nursery school application: 12,960 records, 8 categorical attributes, 4 classes
  - Census: 32,561 records, 14 discrete or categorical attributes, 2 classes
  - Attributes are encoded as integers before fingerprinting
- Baselines

baseline (i)	data perturbation followed by fingerprinting	two-step
baseline (ii)	data synthesis followed by fingerprinting	two-step
baseline (iii)	k-anonymity-based fingerprinting	two-step
baseline (iv)	privacy-protection fingerprinting via Gaussian noise by Hu et al.	one-step
baseline (v)	data perturbation only via local differential privacy	no liability
baseline (vi)	fingerprinting only via mechanism developed by Li at al.	no privacy

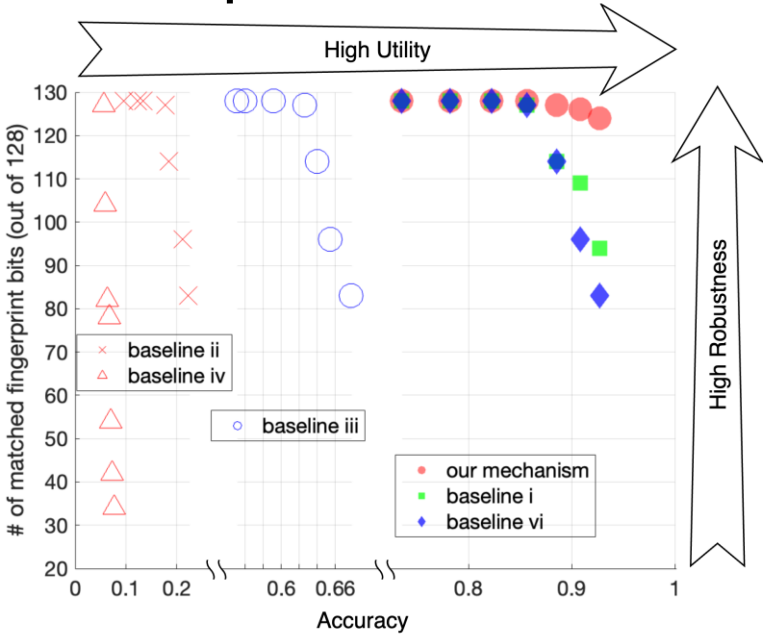
Hu et al., "Towards a privacy protection-capable noise fingerprinting for numerically aggregated data", Computers & Security.
Li et al., "Fingerprinting relational databases: Schemes and specialties", IEEE TDSC.

Experiments

- Use 128 bits for fingerprint and consider 50% random bit flipping attack
- x-axis: accuracy of fingerprinted DB
- y-axis: match of extracted fingerprint from compromised DB

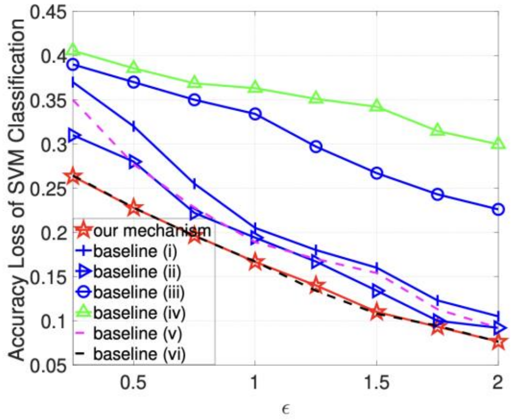


Nursery

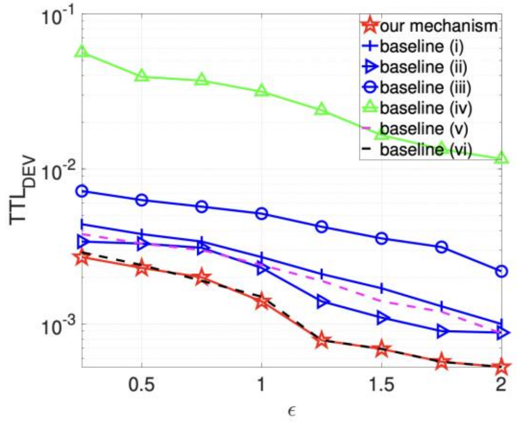


Census

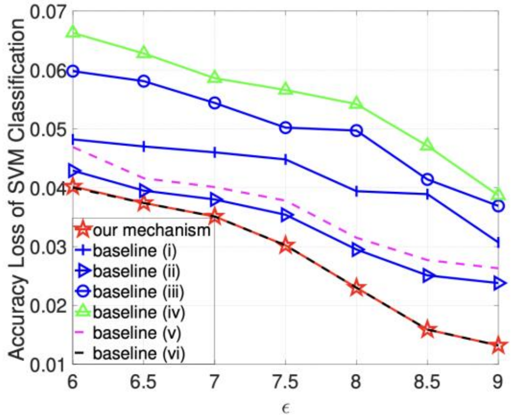
Experiments



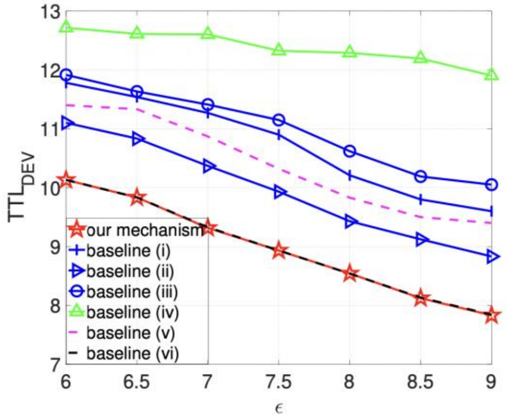
(a) SVM on Nursery Database.



(b) PCA on Nursery Database.



(a) SVM on Census Database.



(b) PCA on Census Database.

Conclusions

- Developed the first privacy-preserving DB fingerprinting scheme
- Connect privacy, fingerprint robustness, and DB utility
- Use SVT to control cumulative privacy loss

- Future work
  - Mitigate correlation attacks
  - Improve utility by utilizing data distribution
  - Defend against membership inference attack

Contact: Tianxi Ji
tiji@ttu.edu

Entry-level DP v.s. DP

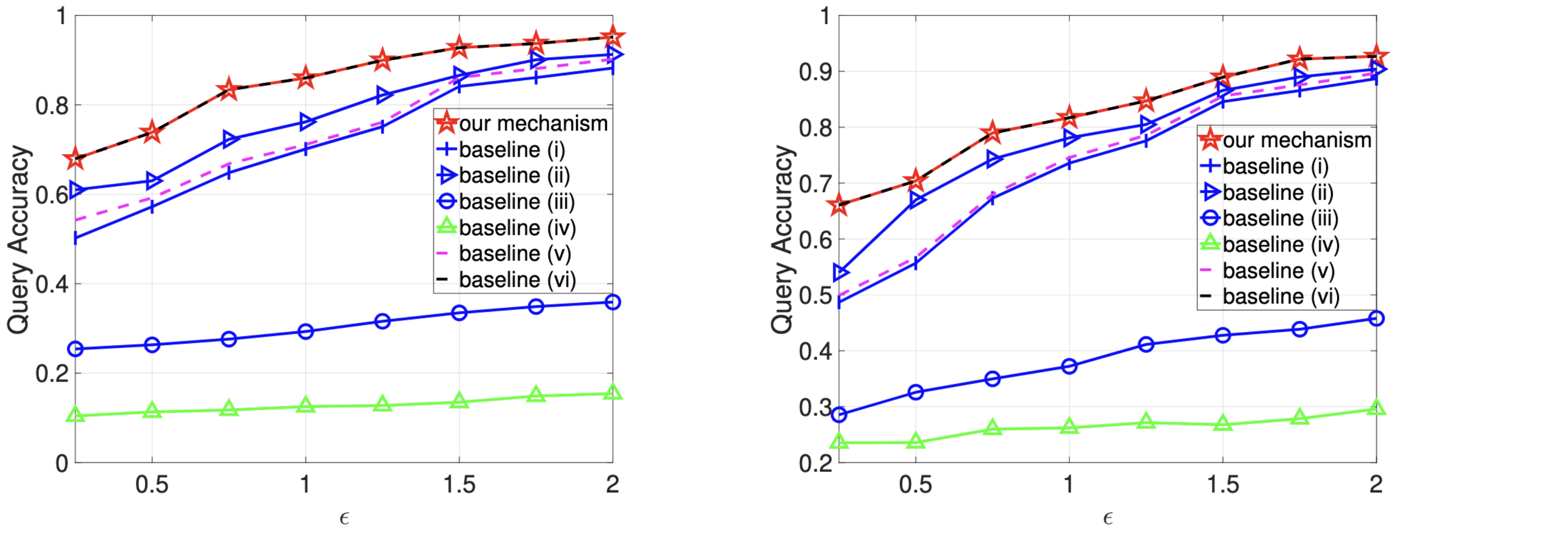
No matter what learning-based inference attack the malicious SP conducts, its inference capability can never be higher than $\frac{\psi e^{\epsilon}}{\psi e^{\epsilon}+1}$, i.e., $\mathrm{InfCap} \leq \frac{\psi e^{\epsilon}}{\psi e^{\epsilon}+1}$, where $\psi = \frac{\Pr(\boldsymbol{r}_i[t]=\zeta_1 \mid \mathbf{R}_{/\boldsymbol{r}_i[t]})}{\Pr(\boldsymbol{r}_i[t]=\zeta_2 \mid \mathbf{R}_{/\boldsymbol{r}_i[t]})}$ is the ratio of the malicious SP's prior knowledge of the unknown entry $\boldsymbol{r}_i[t]$ taking different values (i.e., ζ_1 and ζ_2) given all other entries are known.

- All entries in DB satisfying ϵ-entry-level DP are naturally ϵ-DP for DB
- Privacy amplification occur when ϵ'-DP holds for DB and $\epsilon' < \epsilon$
  - Subsampling
  - Shuffling

DB utility: SQL query

$Q1$:SELECT PmyKey FROM Nursery WHERE
children = more AND social = slightly_prob

$Q2$:SELECT PmyKey FROM Nursery WHERE
parent = usual AND finance = incov

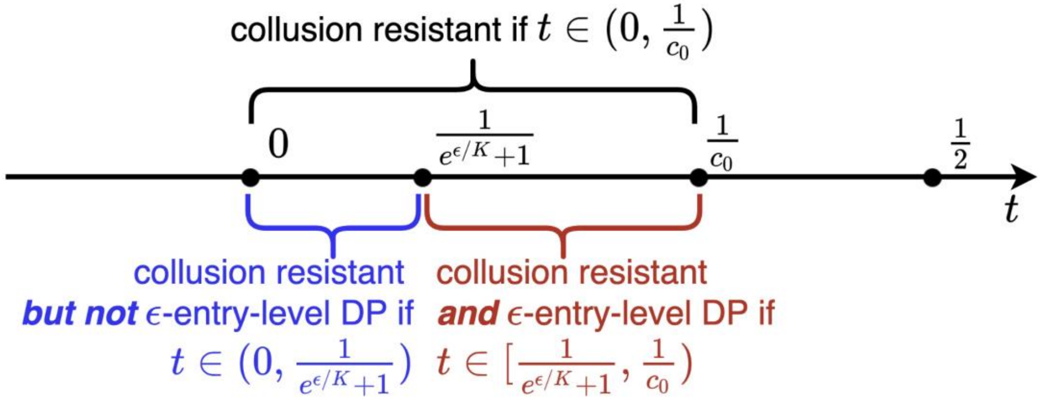


(a) Accuracy of $Q1$.　　(b) Accuracy of $Q2$.

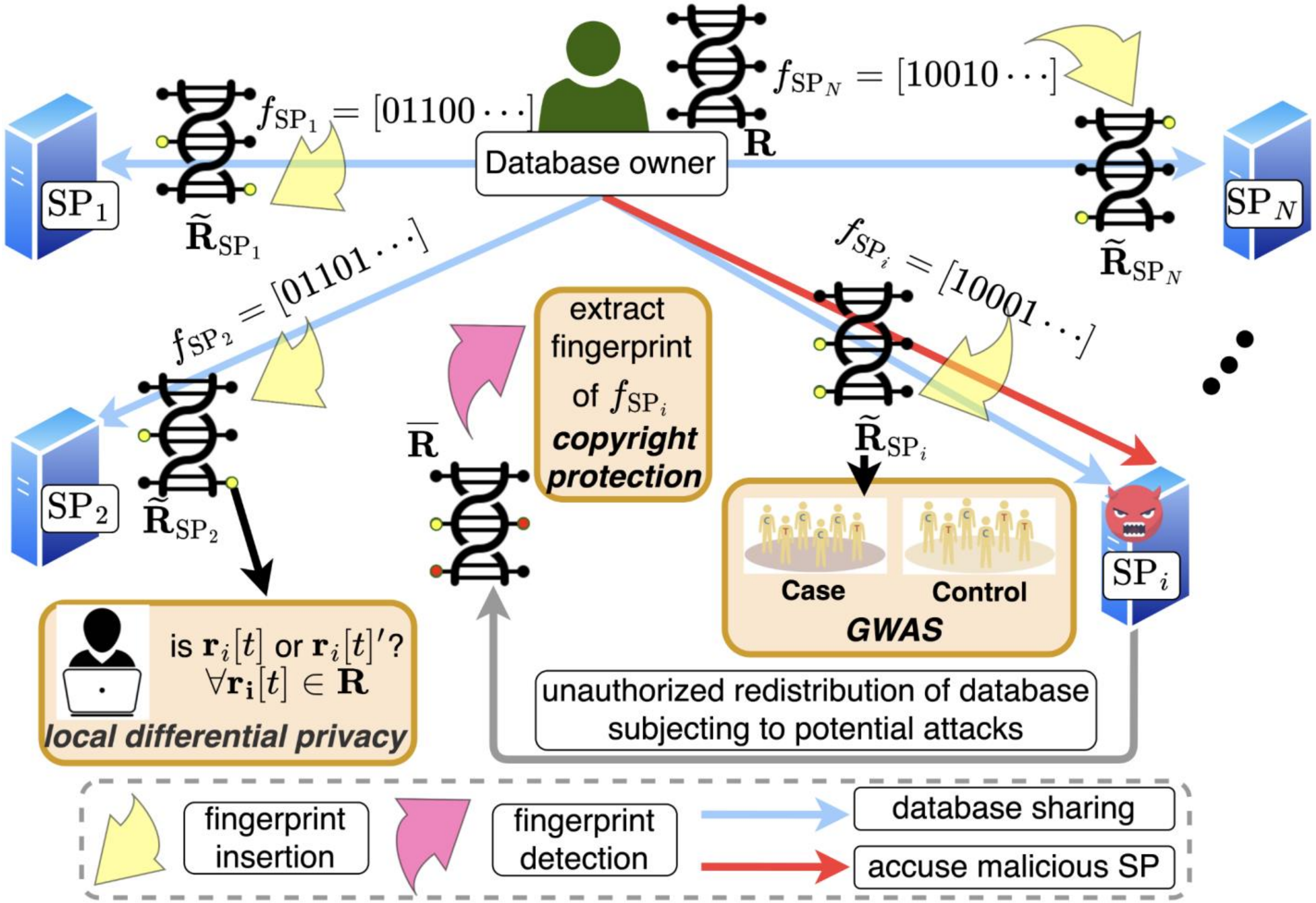
Collusion attack

- Malicious SPs combine their versions of fingerprinted DBs to forge a pirated copy with the hope that none of them can be traced back
- Achieve collusion-resistant, privacy-preserving fingerprinting by leverage randomness of Tardos code

1. Sample a random variable p from probability density function $f(p|t) = \frac{1}{2\arcsin(1-2t)} \frac{1}{\sqrt{p(1-p)}}, t \in (0, 0.5)$.
2. Generate the Tardos fingerprint string, i.e., $\mathbf{f} \sim Bernoulli(p)$.



Application on Genomic DB



https://github.com/xiutianxi/ldp_genomic_fp