Distributed Synthesis of Differentially Private Tabular Datasets

35th USENIX Security Symposium 2026

Presenter: Yucheng Fu

Yucheng Fu [1], Tianyao Gu [2], Elaine Shi [2], Tianhao Wang [1]

[1] University of Virginia [2] Carnegie Mellon University

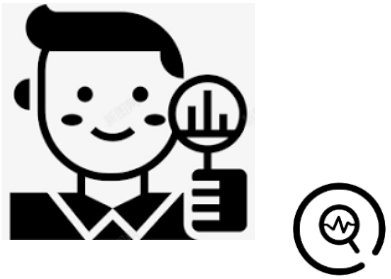
Differentially Private Dataset Synthesis



Hospital

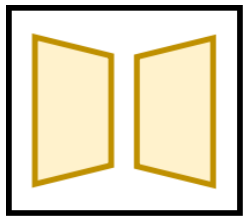
Age	Blood	Disease
34	A+	Asthma
58	O	Diabetes
42	AB+	Hypertension

Medical record



Researchers

Satisfies DP guarantee

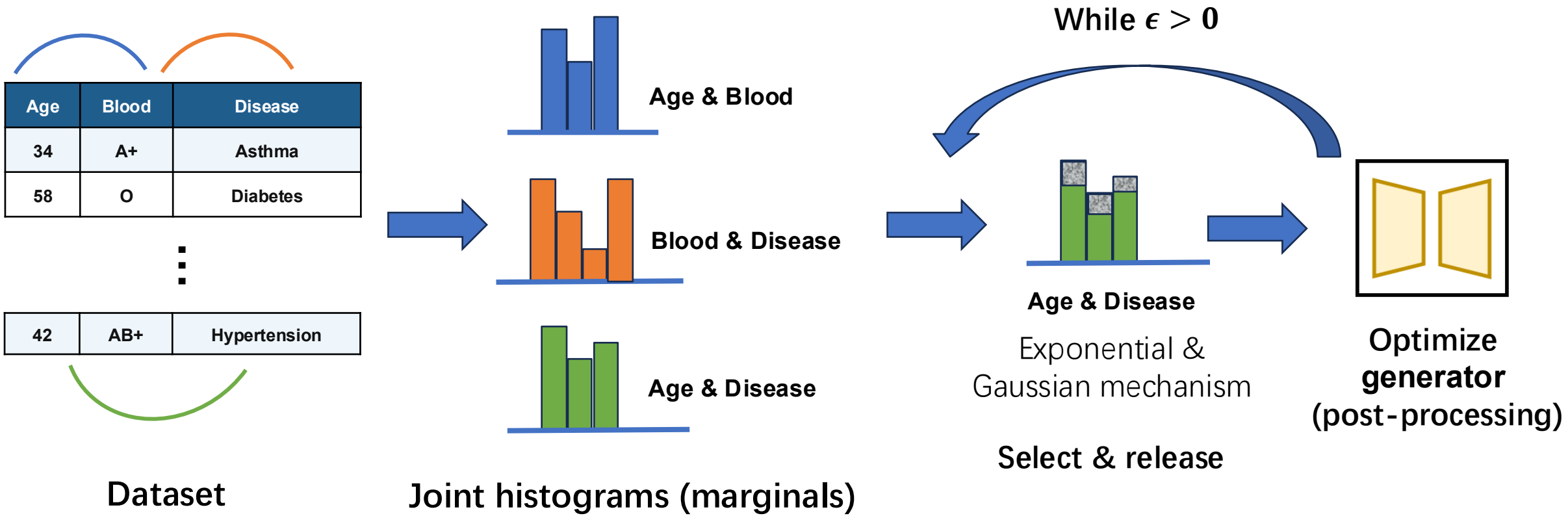


Generator

Age	Blood	Disease
35	A+	Asthma
56	B	Diabetes
45	O	Hypertension

Synthetic data

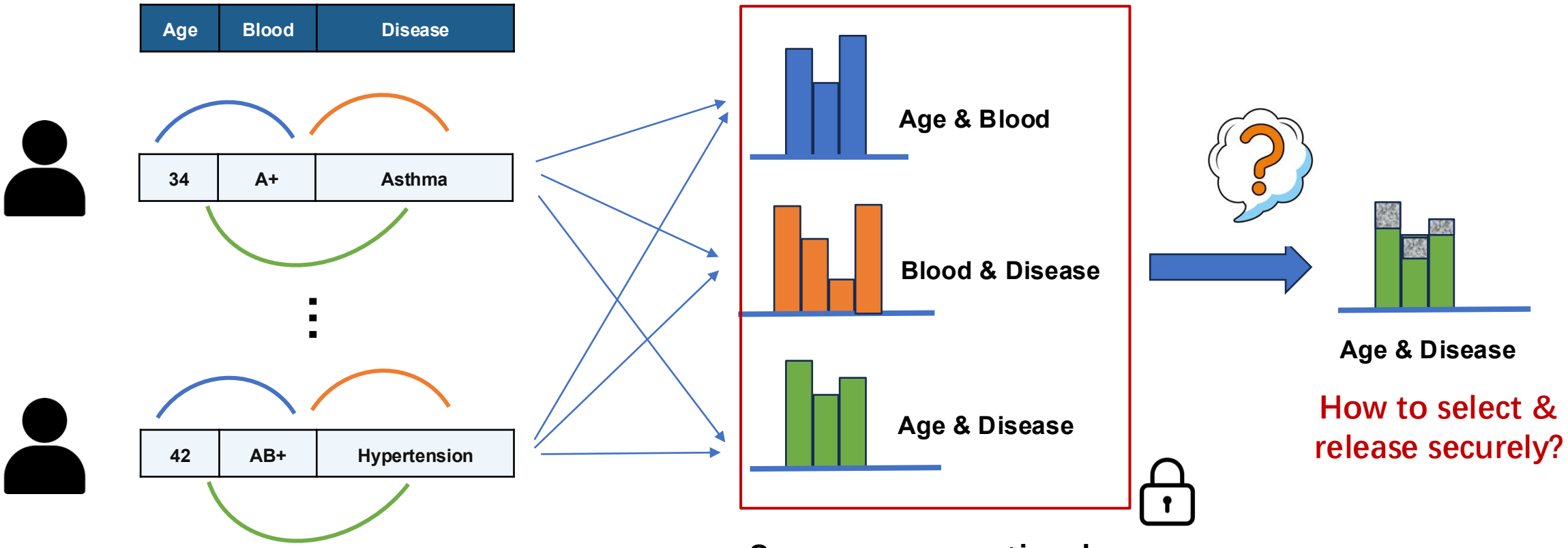
Centralized Differentially Private Synthesis[a,b]



[a] Ryan McKenna, et al. "AIM: An Adaptive and Iterative Mechanism for Differentially Private Synthetic Data." VLDB, 2022.

[b] Ryan McKenna, et al. "Graphical-model based estimation and inference for differential privacy." ICML, 2019,

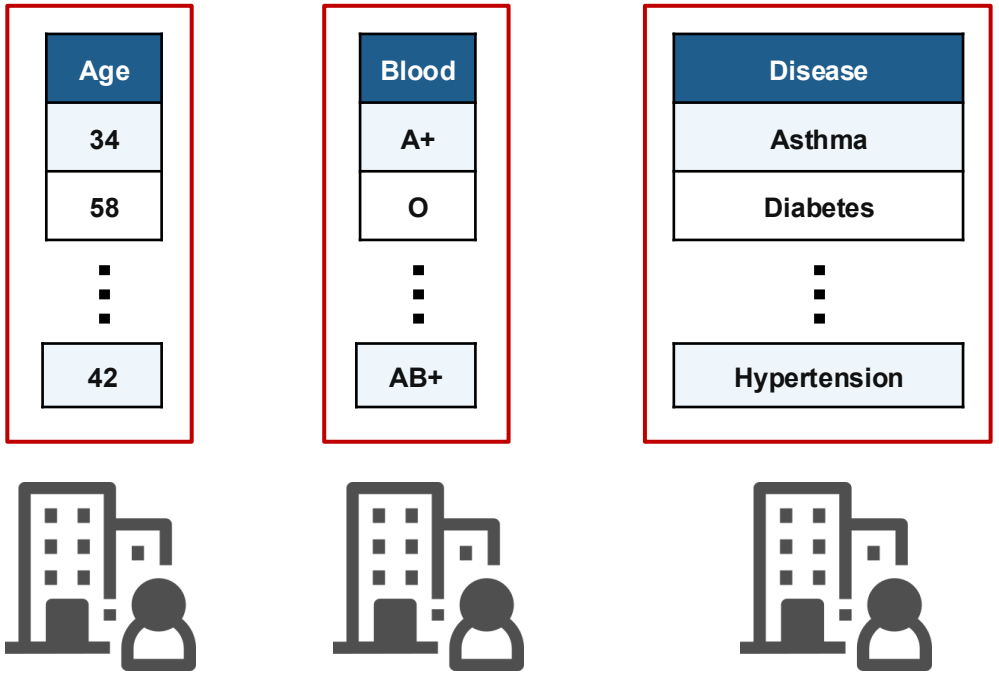
Differentially Private Synthesis in a Distributed Setup



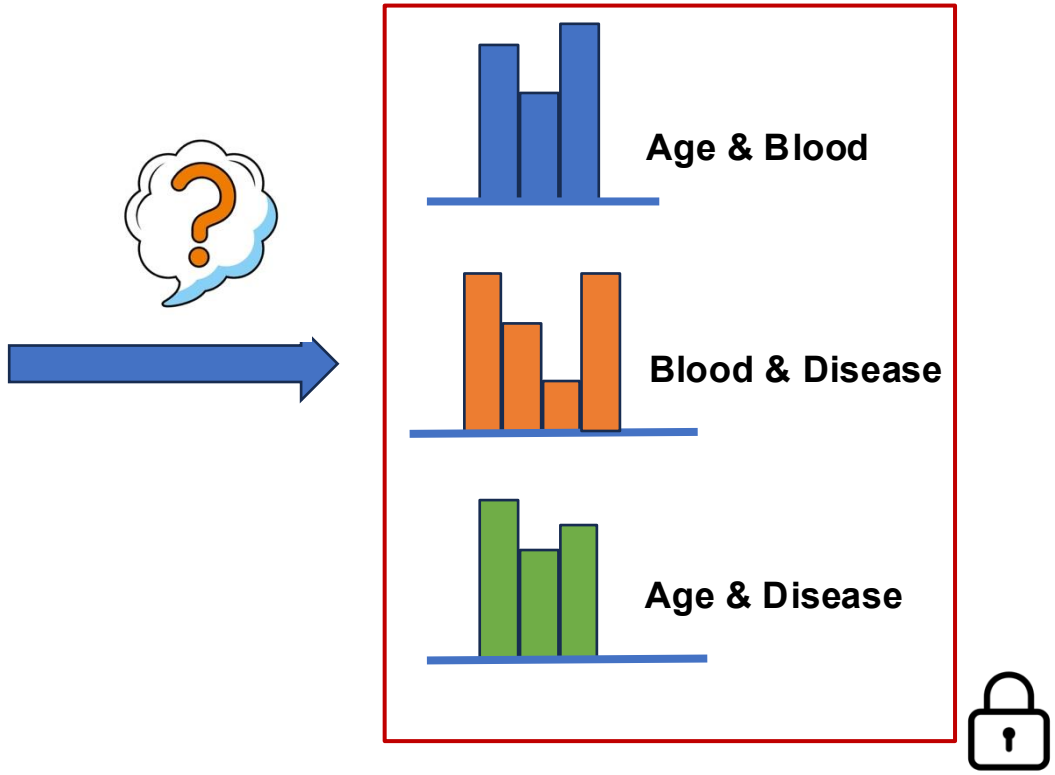
Horizontally distributed dataset

Secure aggregation by sub-strings

Differentially Private Synthesis in a Distributed Setup



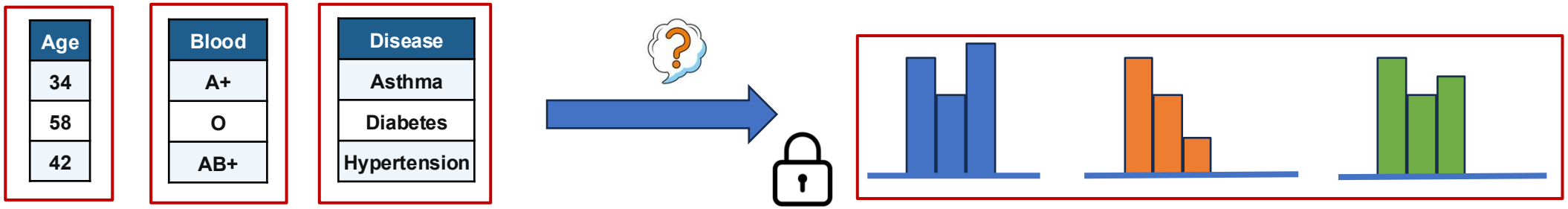
Vertically distributed dataset



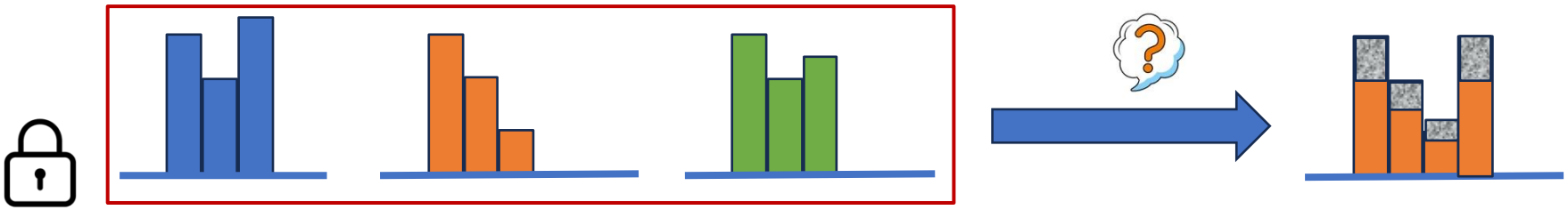
How to securely compute marginals?

Challenges

1. How to securely and efficiently compute the marginals among vertically distributed columns from different entities?

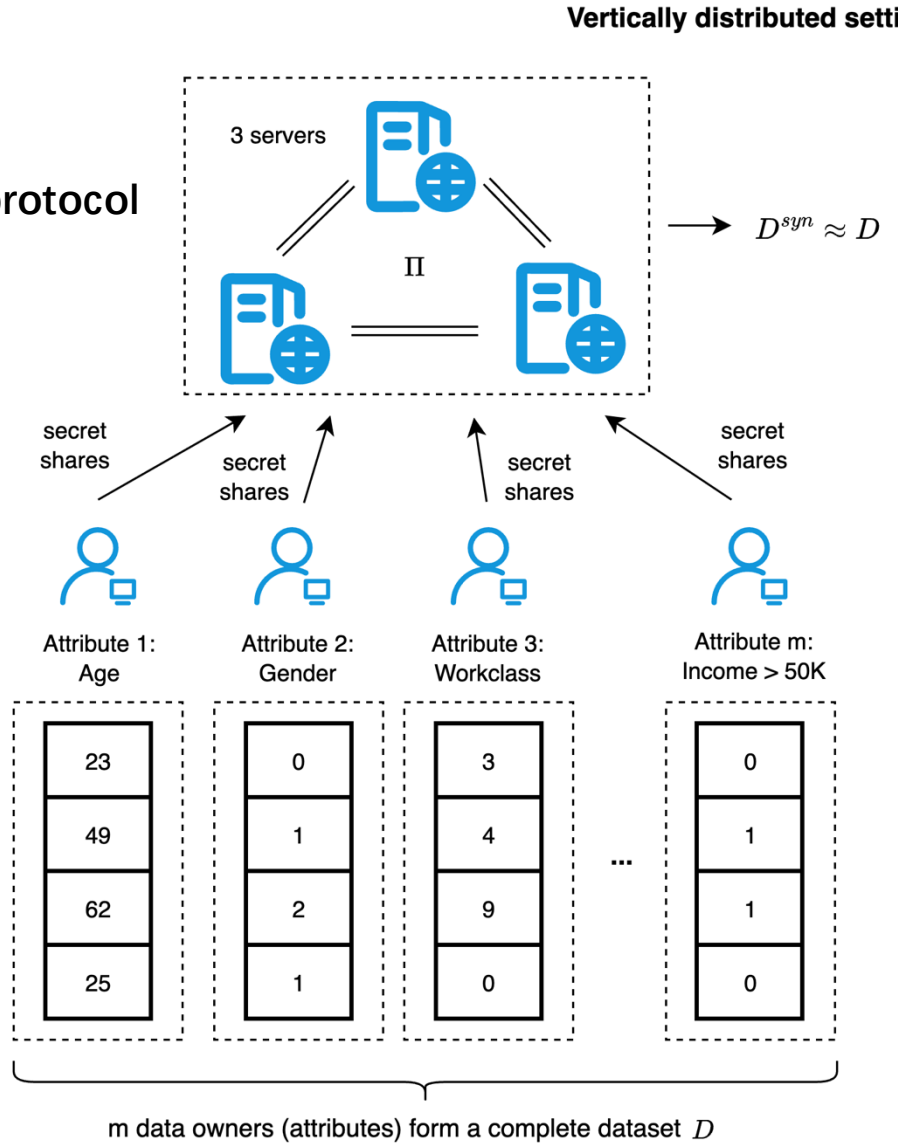


2. After getting the secret marginals, how to securely and efficiently select & release with DP guarantee?



Problem Definition

A secure multi-party computation (MPC) protocol

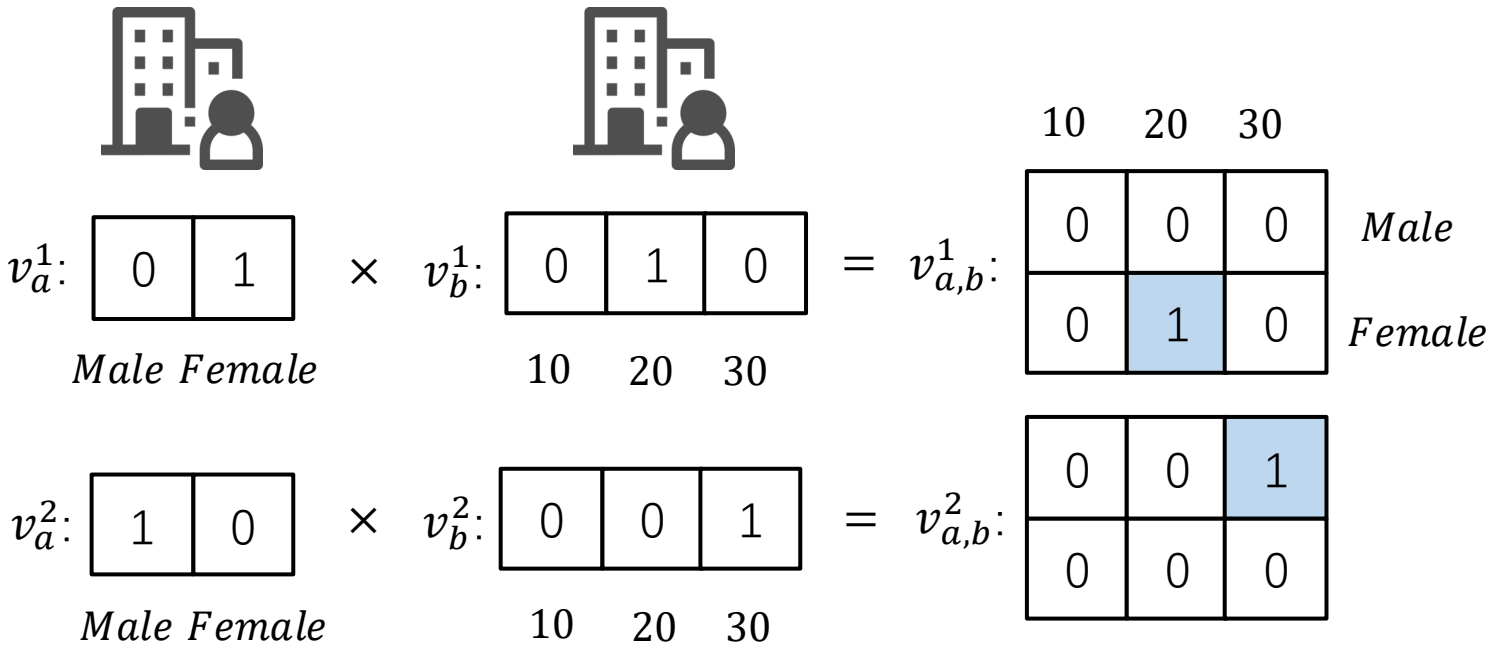


- m data owners (entities), each holding a single column, and three non-colluding servers.
- The data owners hope to protect their private columns against other owners/servers.

Data owners secret-share their private columns to the servers, who run a computationally differentially private protocol.

Component 1: Secure Marginal Computation

Intuition: the marginal of two columns can be obtained by adding the outer products of one-hot encodings.



$\mathrm{M}_{a,b} = \sum_{i=1}^{n} v_{a,b}^i$:

20	5	10
15	10	40

Addition can be done locally for secret-shared $v_{a,b}^1 \ldots v_{a,b}^n$

Component 1: Secure Marginal Computation

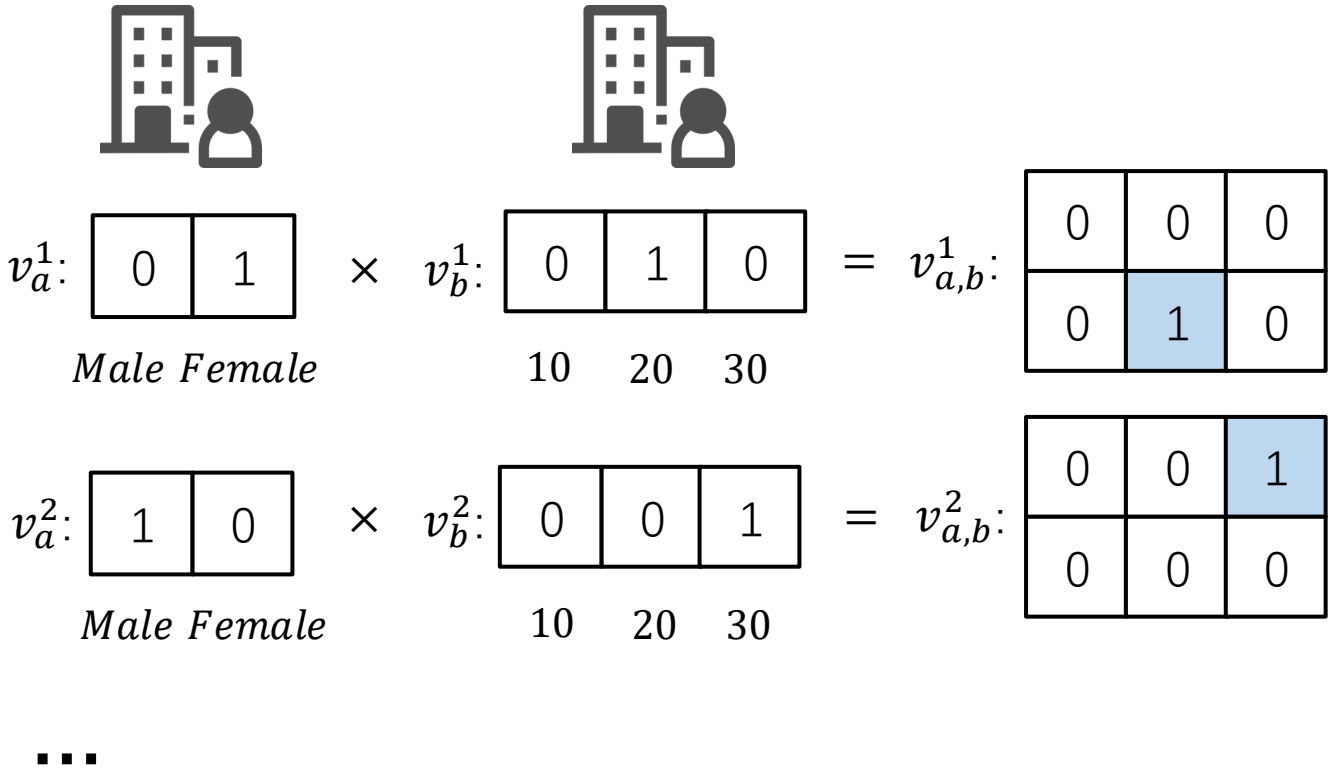
Problem:

1. Secret sharing a one hot encodings v_a^1 of size u is expensive:

$O_\lambda(u)$ communication

2. Compute the outer-product for each pair of v_a^i and v_b^i is expensive :

u^2 times of multiplication for secret shares ⟶ $O_\lambda(u^2)$ communication



Component 1: Secure Marginal Computation

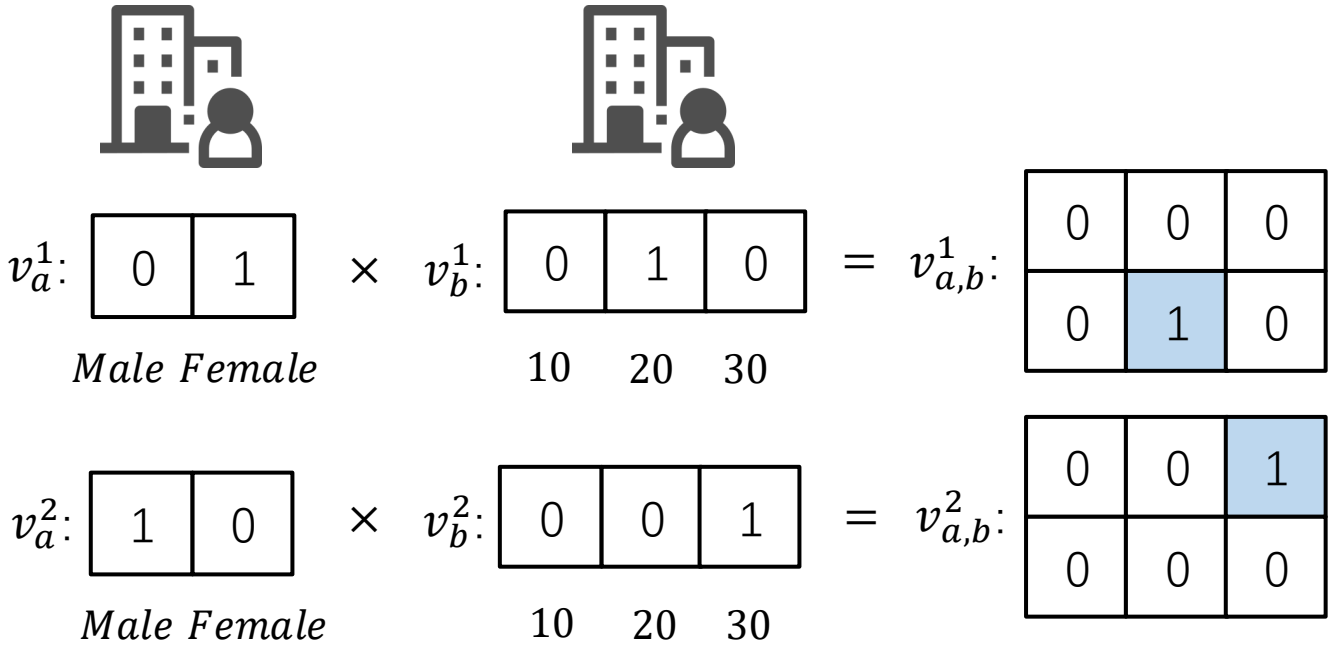
Problem:

1. Secret sharing a one hot encodings v_a^1 of size u is expensive:

~~$O_\lambda(u)$~~ **communication**
$O_\lambda(\log u)$

2. Compute the outer-product for each pair of v_a^i and v_b^i is expensive :

u^2 times of multiplication for secret shares ⟶ ~~$O_\lambda(u^2)$~~ **communication**
0



...

Component 1: Secure Marginal Computation

Use multiplicative distributed point function (DPF) [a] to compress one-hot encoding and enable local multiplication

A multiplicative DPF parameterized by $\mathbb{G}_{in}$ is a pair of randomized algorithms $\mathbf{Gen}, \mathbf{Eval}$:

$k_1, k_2, k_3 \leftarrow \mathrm{Gen}^{\mathbb{G}_{in}}(1^\lambda, \mathrm{a})$, $|k_i| = \log |\mathbb{G}_{in}|$

$s_1, s_2 \leftarrow \mathrm{Eval}(1^\lambda, k_1, x)$

$s_2, s_3 \leftarrow \mathrm{Eval}(1^\lambda, k_2, x)$ such that $s = \sum s_i = \begin{cases} 1 & if\ x = \alpha \\ 0 & else \end{cases}$

$s_3, s_1 \leftarrow \mathrm{Eval}(1^\lambda, k_3, x)$

$\langle s \rangle = \{ (s_1, s_2), (s_2, s_3), (s_3, s_1) \}$ are replicated secret shares that tolerates 1 corruptions of 3 parties

[a] Amit Agarwal, et al. "Compressing Unit-Vector Correlations via Sparse Pseudorandom Generators." CRYPTO, 2024.

Component 1: Secure Marginal Computation

Use multiplicative distributed point function (DPF) [a] to compress one-hot encoding and enable local multiplication

A multiplicative DPF parameterized by $\mathbb{G}_{in}$ is a pair of randomized algorithms $\mathrm{Gen}, \mathrm{Eval}$:

$k_1, k_2, k_3 \leftarrow \mathrm{Gen}^{\mathbb{G}_{in}}(1^\lambda, \mathrm{a})$, $|k_i| = \log |\mathbb{G}_{in}|$

$s_1, s_2 \leftarrow \mathrm{Eval}(1^\lambda, k_1, x)$

$s_2, s_3 \leftarrow \mathrm{Eval}(1^\lambda, k_2, x)$ such that $s = \sum s_i = \begin{cases} 1 & if\ x = \alpha \\ 0 & else \end{cases}$

$s_3, s_1 \leftarrow \mathrm{Eval}(1^\lambda, k_3, x)$

$\langle s \rangle = \{ (s_1, s_2), (s_2, s_3), (s_3, s_1) \}$ are replicated secret shares that tolerates 1 corruptions of 3 parties

Local multiplication of replicated secret shares:

Party i holds: (a_i, a_{i+1}) and (b_i, b_{i+1})

Computes: $c_i = a_i b_i + a_i b_{i+1} + a_{i+1} b_i$

Since $\sum_i c_i = (a_1 + a_2 + a_3)(b_1 + b_2 + b_3) = ab$

party i now has the additive share of ab

[a] Amit Agarwal, et al. "Compressing Unit-Vector Correlations via Sparse Pseudorandom Generators." CRYPTO, 2024.

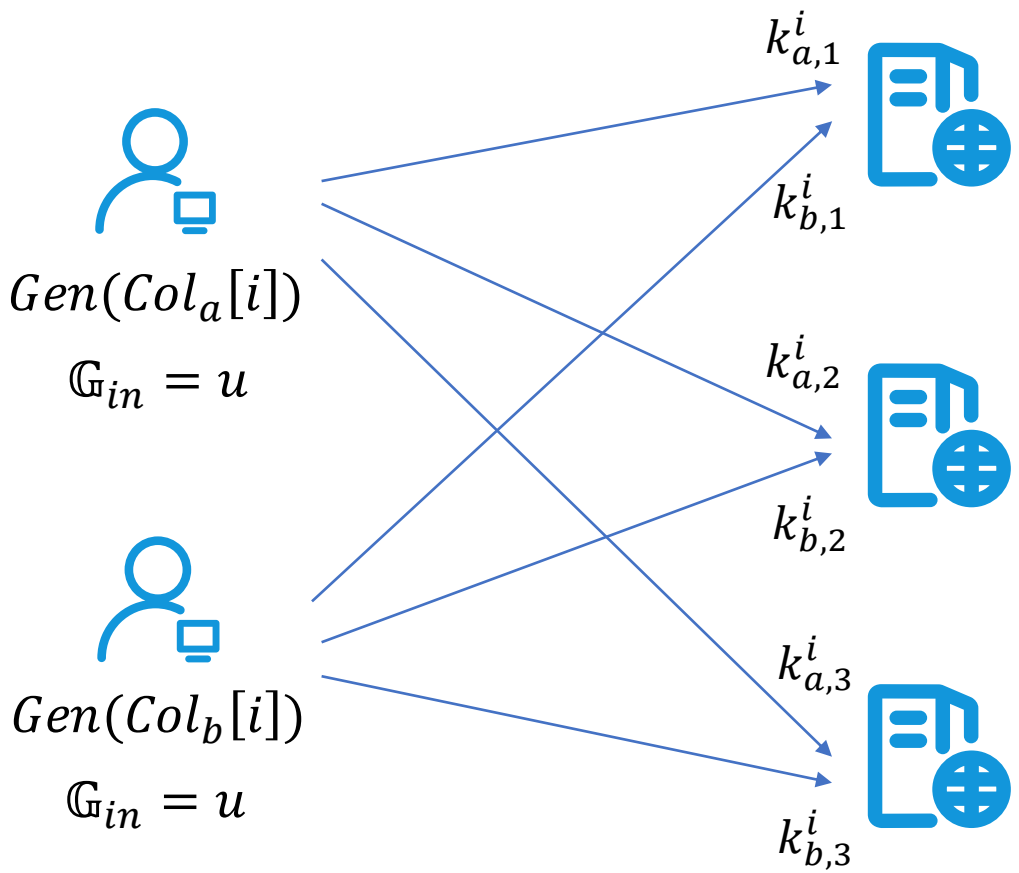
Component 1: Secure Marginal Computation

$O_\lambda(log\, u)$ PRG calls

$O_\lambda(log\, u)$ communication

$O_\lambda(u)$ PRG calls

Locally multiplying replicated shares



Component 2: Secure Discrete Noise Generation

Target: generate d secret shared samples $\langle \eta_1 \rangle, \ldots, \langle \eta_d \rangle$ from discrete distribution (e.g., discrete Gaussian, Geometric)

Existing methods to sample discrete noise for computational differential privacy [a,b,c,d]



Discrete Laplace (Geometric) noise Discrete Gaussian noise

Limitation 1: not optimized when generating multiple noise samples.

Limitation 2: primarily work in GC, which incurs large round complexity in the secret sharing protocols.

Our starting point: make a direct lookup in the CDF table for the discrete distribution.

[d] Champion, et al. "Securely Sampling Biased Coins with Applications to Differential Privacy." CCS, 2019.
[a] Chengkun Wei, et al. "Securely Sampling Discrete Gaussian Noise for Multi-Party Differential Privacy." CCS, 2023.

Component 2: Secure Discrete Noise Generation

A strawman secure protocol:
Suppose discrete Gaussian distribution $\mathcal{N}$ is truncated into $[-s, s]$, and F is its cummulative distribution function.

1. Publicly compute a table T indexed by the CDF to store all possible values in $[-s, s]$.

-s	⋯	-1	0	1	⋯	s
$F(-s)$	⋯	$F(-1)$	$F(0)$	$F(1)$	⋯	$F(s)$

2. Sample a secret-shared random number $\langle r \rangle$ in [0,1] and conduct a linear scan on T.

3. Return $\langle i \rangle$ such that $F(i) \leq r \leq F(i+1)$

$O(s)$ communication per sample

Component 2: Secure Discrete Noise Generation

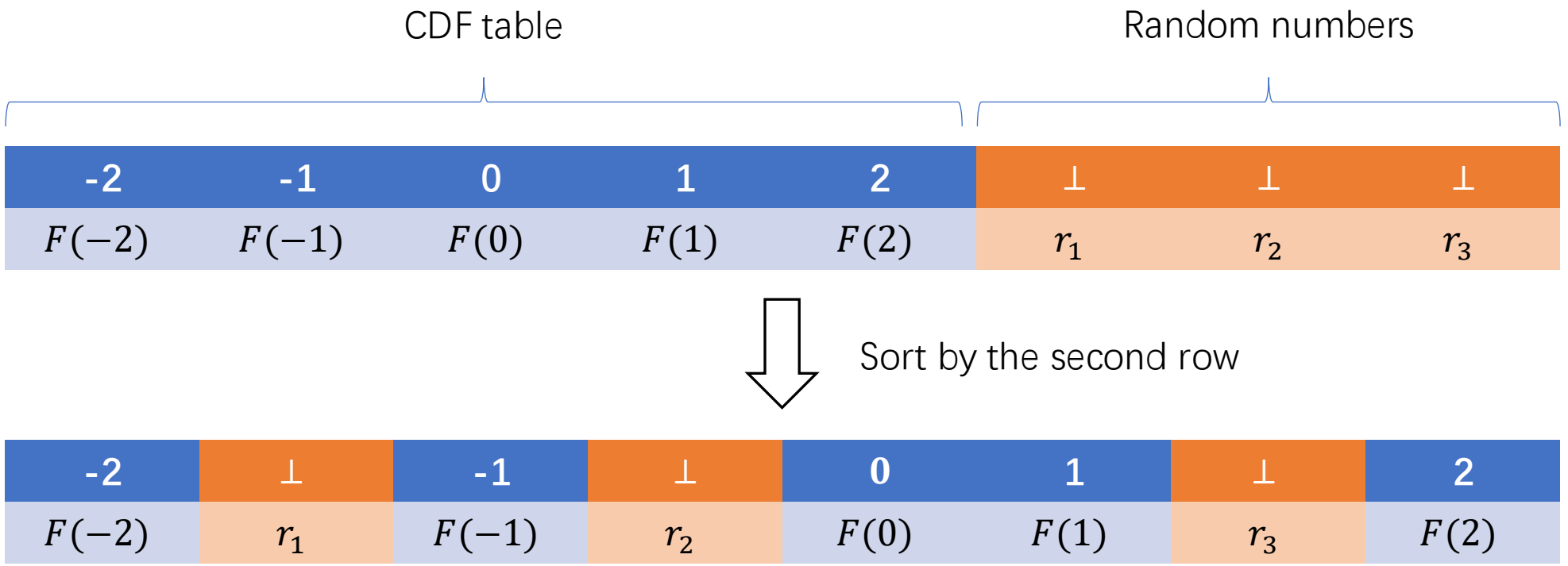
Observation 1: using sorting to achieve batched table lookup

- Reduce communication complexity

Observation 2: using a parallel segmented scan to extract the samples

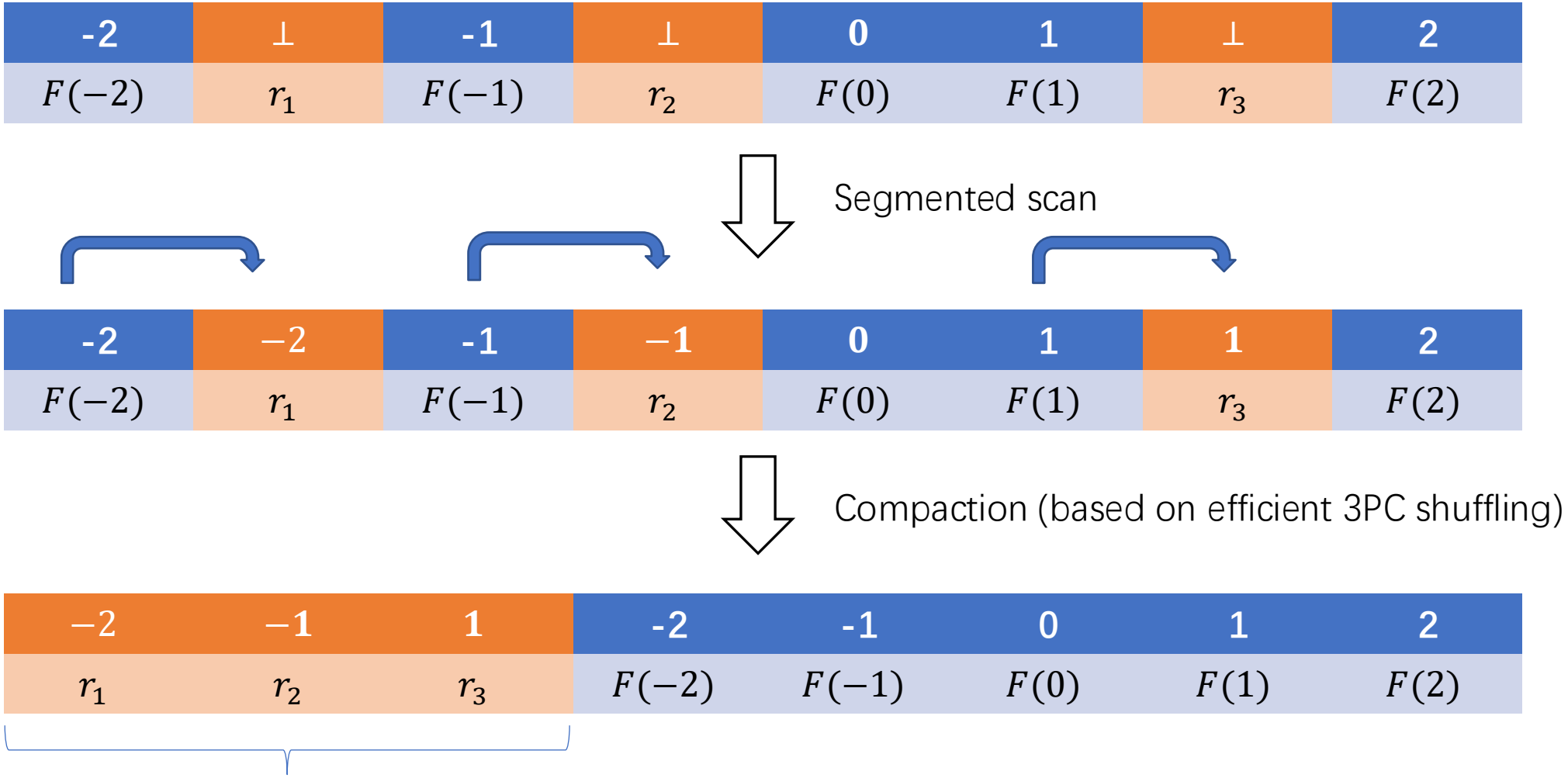
- Reduce round complexity

Component 2: Secure Discrete Noise Generation



After sorting, the random numbers are located among the CDFs; we need to convert and extract them

Component 2: Secure Discrete Noise Generation



Putting Things Together

Age
34
58
42

Blood
A+
O-
AB+

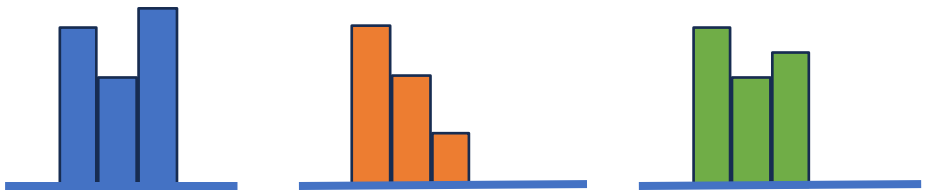
Disease
Asthma
Diabetes
Hypertension

Noise generation protocol

Marginal computation protocol

Geometric noise $G_{1,2}, G_{2,3}, G_{1,3}$

Discrete Gaussian noise $\mathcal{N}$



$$q_{1,2} = \left\| \mathrm{M}_{1,2} - \widetilde{\mathrm{M}}_{1,2} \right\|_1 + G_{1,2}$$

$$q_{2,3} = \left\| \mathrm{M}_{2,3} - \widetilde{\mathrm{M}}_{2,3} \right\|_1 + G_{2,3}$$

$$q_{1,3} = \left\| \mathrm{M}_{1,3} - \widetilde{\mathrm{M}}_{1,3} \right\|_1 + + G_{1,3}$$

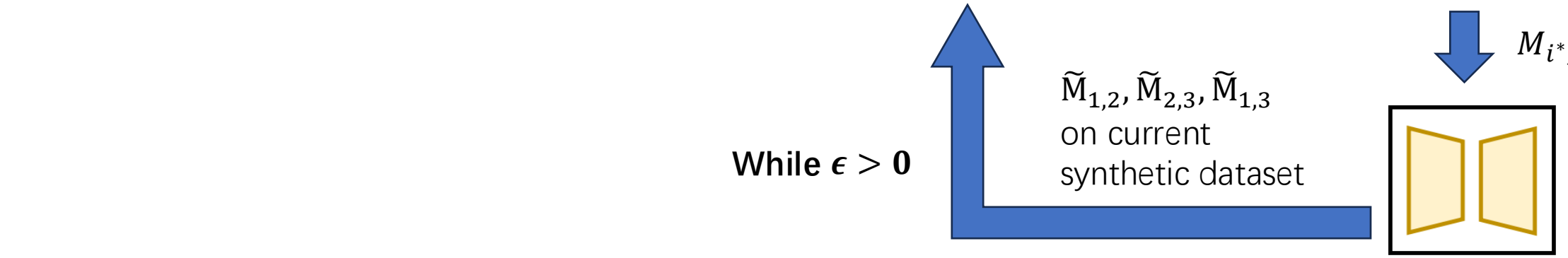
$$\mathrm{M}_{i^*,j^*} = \mathrm{M}_{i^*,j^*} + \mathcal{N}$$

$$i^*, j^* = \mathrm{argmax}(q_{i,j})$$

M_{i^*,j^*}

$\widetilde{\mathrm{M}}_{1,2}, \widetilde{\mathrm{M}}_{2,3}, \widetilde{\mathrm{M}}_{1,3}$ on current synthetic dataset

While $\epsilon > 0$

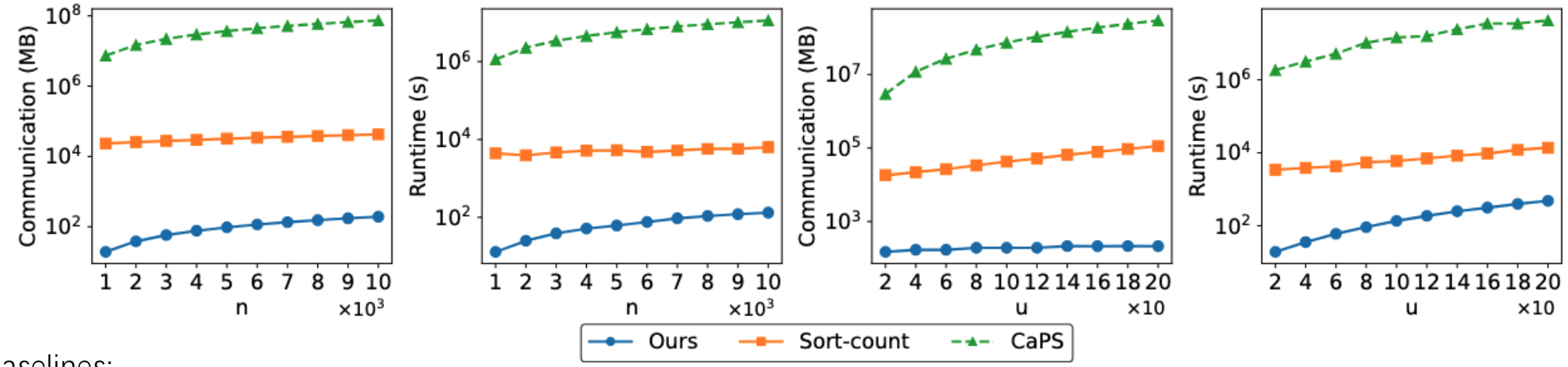


Evaluation Setup

- **Hardware**: c5a.8xlarge AWS instance with 64 GB RAM and 32 vCPUs.
- **Network:** clients and three servers in a WAN setting, with bandwidth up to 0.5 Gbit/s and a latency of 50 ms.
- **Efficiency metrics:**
  - Global communication
  - Number of communication rounds
  - Wall-clock runtime
- **Utility metrics (using existing benchmark) [a] :**
  - Accuracy of machine learning model trained on the synthetic dataset
  - SQL query error between the synthetic dataset and the real dataset

[a] Kai Chen, et al. "Benchmarking Differentially Private Tabular Data Synthesis:[Experiments & Analysis]." *Proceedings of the ACM on Management of Data* 3.6 (2025): 1-25.

Evaluation: Joint Histogram (Marginal) Computation



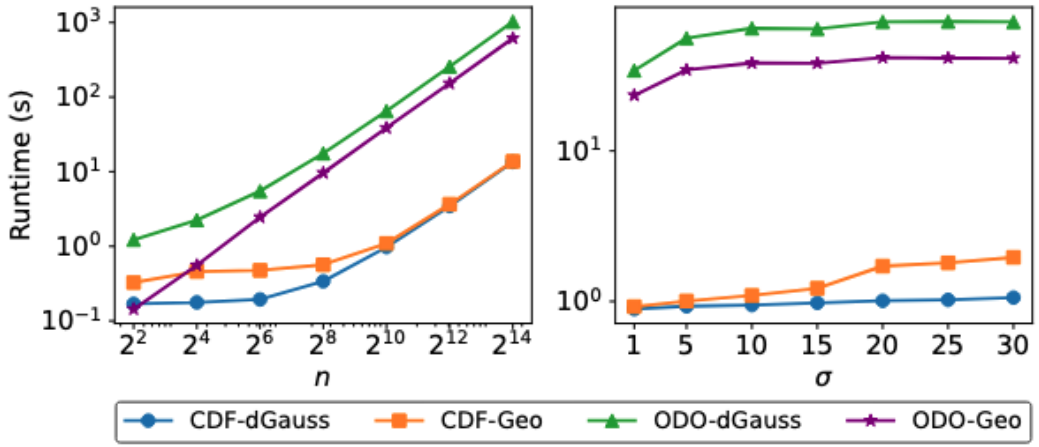
Baselines:

1. CaPS [a]: existing protocol for distributed DP data synthesis
2. Sort-count [b]: alternative protocol for joint histogram computation

1. **Our method takes ≈ 2 minutes to compute a marginal of size 100×100 on a table with 10^4 records.**
2. **Five-order-of-magnitude faster than existing data synthesis protocol and $30\times\sim 290\times$ faster than alternative protocol.**

[a] Sikha Pentyala, et al. "CaPS: Collaborative and Private Synthetic Data Generation from Distributed Sources." ICML, 2024.
[b] Gilad Asharov, et al. "Secure Statistical Analysis on Multiple Datasets: Join and Group-By." CCS, 2023

Evaluation: Noise Generation



Overhead	CDF-dGauss	CDF-Geo	ODO-dGauss	ODO-Geo
Time (s)	13.41	14.05	932.65	535.49
Comm (MB)	1998.6	2128.32	76.10	44.69
Rounds	4,234	4,342	12,017,375	7,159,845

Baseline: bitwise sampling for discrete Gaussian [a] and Geometric [b]

1. **Our CDF-based protocol has a significantly lower runtime under various parameter settings.**
2. **Very low round complexity compared to the existing noise generation protocol.**

[a] Chengkun Wei, et al. "Securely Sampling Discrete Gaussian Noise for Multi-Party Differential Privacy." CCS, 2023.
[b] Champion, et al. "Securely Sampling Biased Coins with Applications to Differential Privacy." CCS, 2019.

Evaluation: End-to-end synthesis

Overhead	Total	Marginals	ABS	Noise	Argmax	Generate
Time (min)	24.12	2.37	8.21	6.27	0.08	7.19
Comm (MB)	44946.51	542.28	21289.89	12.45	787.87	—
Rounds	1,037,871	1	857,710	132,805	47,355	—

Utility metric	Method	$\varepsilon = 0.1$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 10$
Query Error	AIM [3]	10.92%	7.06%	4.65%	1.72%	1.80%
	Ours	10.59%	7.75%	4.52%	2.08%	1.84%
ML Accuracy	Non-DP	86.47%				
	AIM [3]	82.17%	82.87%	85.07%	85.42%	85.90%
	Ours	81.66%	83.39%	84.96%	85.38%	85.77%

Dataset: "Adult" dataset with 39,073 records and 6×10^{17} Cartesian product domain size

1. **End-to-end synthesis in ≈ 24 minutes vs ≈ 57 days for existing distributed synthesis [a].**
2. **Comparable utility for downstream tasks as the SOTA centralized algorithm, AIM [b].**

[a] Sikha Pentyala, et al. "CaPS: Collaborative and Private Synthetic Data Generation from Distributed Sources." ICML, 2024.
[b] Ryan McKenna, et al. "AIM: An Adaptive and Iterative Mechanism for Differentially Private Synthetic Data." VLDB, 2022.

Contributions and Future Work

Contributions:

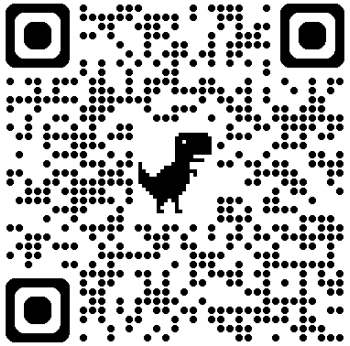
- We built an MPC protocol for distributed synthesis of differentially private tabular datasets, achieving orders-of-magnitude speedup compared to the existing approach.
- We propose a concretely efficient protocol for joint histogram computation, which serves as a key building block to accelerate the distributed dataset synthesis.
- We propose a new protocol for sampling random numbers from discrete distributions such as the Geometric and discrete Gaussian distributions.

Future Work:

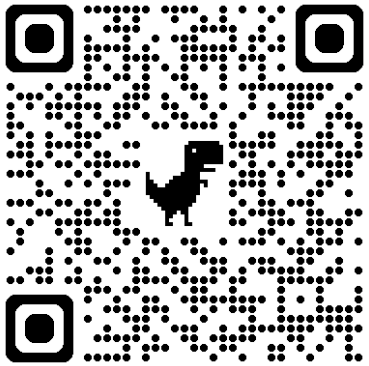
- **Dishonest-majority setting:** Adapting recent advanced techniques in multi-party DPF to support a stronger adversarial model.
- **Potential applicability to other fields:** Our building blocks for joint histogram computation and discrete noise generation may be of independent interest and could be useful in broader privacy-preserving analytics and secure computation tasks.

Hope it inspires!

Questions are welcome 😊!



Paper



Artifact