Geometric Garbling: Efficient Two-Party Computation for Differentially Private Mechanisms

Arisa Tajima
UMass Amherst
atajima@cs.umass.edu

Adam O'Neill
UMass Amherst
adamo@cs.umass.edu

Wei Jiang
Oracle Labs
wei.wj.jiang@oracle.com

Gerome Miklau
UMass Amherst
miklau@cs.umass.edu

Abstract

We study privacy-preserving data collaboration in which a platform offers differentially private (DP) query services using proprietary optimization strategies, enabling end users to obtain query answers over a curator's sensitive data. This setting requires secure two-party computation to protect both the curator's data and the platform's strategy parameters. Existing garbled circuit-based protocols provide strong cryptographic security but incur substantial communication and computation overhead by applying cryptographic protection to every intermediate computation.

Our key observation is that full cryptographic secrecy of intermediate values is unnecessary when the released outputs are already differentially private. Leveraging this insight, we introduce a DP-based garbling scheme that replaces cryptographic protection of intermediate wires in a circuit with DP mechanisms. We develop efficient garbling techniques for matrix multiplication, a fundamental primitive underlying a broad class of DP query-answering mechanisms. By carefully accounting for privacy composition across intermediate computations and final outputs, we preserve the same overall DP guarantee while significantly reducing cryptographic overhead. Our experimental results show that our approach achieves up to 3× speedup and 15× reduction in network traffic compared to state-of-the-art garbling techniques, while maintaining accuracy comparable to a non-private baseline.

1 Introduction

Differential privacy (DP) has become the standard for protecting individual privacy in data publication and statistical analysis, with adoption by major organizations including Google, Apple, and the US Census Bureau [3, 20, 1]. As data collection intensifies and privacy regulations such as the European Union's General Data Protection Regulation [21] and the United States' Health Insurance Portability and Accountability Act of 1996 [2] impose strict compliance requirements, DP deployment has become increasingly essential across healthcare, finance, and government sectors.

However, effective DP deployment requires substantial expertise in mechanism design and optimization. Many organizations lack the in-house capabilities to carefully tune DP mechanisms for complex workloads. This motivates a collaboration model in which external analytics platforms provide optimization expertise while organizations retain control over sensitive data.

This model introduces two privacy requirements. Individual privacy must be protected through DP, ensuring that released statistics do not reveal information about specific individuals in the database. At the same time, the platform's strategy parameters must remain confidential as intellectual property. For example, a platform may design optimized measurement strategies for given query workloads by carefully selecting which measurements to compute and how to weight them to maximize accuracy under privacy constraints [35, 38]. These specific parameter values constitute the proprietary asset that the platform wishes to protect.

This workflow requires secure two-party computation [25], in which the data curator and the platform jointly compute DP outputs while protecting both the curator's sensitive data and the platform's proprietary

strategies. Garbled circuits [46] (aka. garbling schemes [15]), a core primitive of secure two-party computation, achieve this by encoding the computation as an encrypted circuit that can be evaluated without revealing intermediate wire values. However, even highly optimized garbling schemes [8, 48, 29] incur overhead proportional to the cryptographic security parameter (e.g., 128 bits) for every wire. As a result, communication and computation costs are often two to three orders of magnitude higher than non-cryptographic implementations.

A seemingly simpler alternative is for the curator to locally add DP noise before sharing data with the platform. However, this prevents the platform from applying its optimized measurement strategy, resulting in suboptimal noise allocation and significantly degraded accuracy.

1.1 Problem Definition

We address secure two party computation of DP mechanisms, which we call *DP-S2PC*. Consider a data curator with sensitive database $\mathbf{x} \in \mathcal{X}$ and a platform provider with proprietary analytical parameters $\mathbf{w} \in \mathcal{Q}$. Both parties jointly compute a DP mechanism $\mathcal{M}(\mathbf{x}, \mathbf{w})$, with the platform receiving the output for publication to end users. Two asymmetric privacy requirements arise naturally from this setting.

Problem 1 (DP-S2PC)**.** Given a DP mechanism $\mathcal{M} : \mathcal{X} \times \mathcal{Q} \to \mathcal{R}$, design a two-party protocol Π between a data curator with input $\mathbf{x}$ and a platform provider with input $\mathbf{w}$ such that:

- Data privacy: The platform's view is ϵ-DP with respect to $\mathbf{x}$, including the received output.
- Strategy privacy: The curator learns nothing about $\mathbf{w}$ beyond the mechanism output in the cryptographic sense.

We instantiate this framework for the matrix mechanism [35], a widely studied approach for answering linear query workloads under DP, including histograms, marginals, range queries, and data cubes. In this setting, the database is represented as a histogram $\mathbf{x} \in \mathbb{N}^n$, and the platform's parameters are represented as a strategy matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$. The core computation is matrix-vector multiplication $\mathbf{Sx}$, which represents query measurements, followed by the addition of DP noise. Our construction, which we will show, is designed to optimize this matrix multiplication.

1.2 Our Contributions

We observe that when a two-party computation produces DP outputs, enforcing full cryptographic secrecy for all intermediate values may be stronger than necessary. Since the released outputs already satisfy DP, intermediate values need only be protected in a manner that ensures a bounded privacy loss, with the overall guarantee following from the DP composition. This observation enables a fundamental redesign of garbled circuit protocols tailored to DP applications.

We introduce *DP garbling schemes*, a new primitive that provides DP guarantees for computing DP outputs instead of standard cryptographic security. Our main technical contribution is *geometric garbling*, an efficient instantiation of this framework specifically designed for a class of matrix multiplication, exemplified by the matrix mechanism. Geometric garbling protects arithmetic circuit wires through geometric noise [24] rather than cryptographic operations, achieving information-theoretic ϵ-DP in the semi-honest model. While tailored for matrix multiplication rather than general arithmetic circuits, this specialization enables significant efficiency gains.

By combining geometric garbling with sender-random oblivious transfer [4], we construct an online-efficient DP-S2PC protocol tailored to our setting, where query workloads are fixed and known in advance. We demonstrate this protocol on statistical query processing using the matrix mechanism, accommodating a broad class of existing DP query answering techniques [35, 47, 38]. We further introduce a post-processing technique that boosts query answer accuracy by combining multiple DP measurements produced as protocol artifacts. This approach fully utilizes the available privacy budget of ϵ and achieves $O(1/\epsilon)$ error.

We conduct experiments on a variety of datasets and query workloads to evaluate the performance of our DP-S2PC protocol. Our approach achieves accuracy comparable to a non-private baseline, while showing a 3× runtime speedup and a 15× reduction in network traffic compared to state-of-the-art garbling schemes. Moreover, our results demonstrate low online communication overhead, incurring only 1-3x overhead relative to the non-private implementation.

2 Efficient DP-S2PC via Geometric Garbling

Standard garbled circuits use cryptographic security to protect all intermediate wire values. When the function output is already DP, this cryptographic security for intermediate values is stronger than necessary. We can replace it with DP guarantees while maintaining equivalent overall privacy. We introduce DP garbling schemes (see Definition B.1), a framework for garbled circuits that provide DP guarantees instead of cryptographic security. Geometric garbling instantiates this framework specifically for matrix multiplication, providing the building blocks needed to efficiently garble circuits. Our complete protocol for the matrix mechanism combines geometric garbling with oblivious transfer, where geometric garbling provides DP for the curator's data and oblivious transfer provides cryptographic security for the platform's query strategy.

2.1 Geometric Garbling: Building Blocks

Geometric garbling is a key building block for efficiently garbling matrix multiplication circuits under DP, motivated by the matrix mechanism [35]. We introduce efficient wire representations, linear gates, and evaluator half-gates.

DP Wire Labels. The core building block of our construction is is the *DP wire label*, which replaces traditional cryptographic wire labels. A wire carrying value $a \in \mathbb{Z}_q$ is encoded as: $L^a = a + r$ where $r \sim \mathsf{Geo}(\Delta/\epsilon)$ is geometric noise calibrated to sensitivity Δ and privacy parameter ϵ. This encoding provides ϵ-DP for the wire value while requiring only $O(\log q)$ bits for wires over $\mathbb{Z}_q$. Under standard 64-bit arithmetic (i.e., $q = 2^{64}$), each DP wire label requires at most 64 bits. In contrast, state-of-the-art garbling techniques require 2048 bits per wire [8]. This compact representation leads to significant reductions in computation and communication.

Free Linear Operations. The linearity of DP wire labels enables extremely efficient evaluation of linear gates, including addition and multiplication-by-constants. These operations require *no communication, no cryptography, and no additional privacy cost*. The evaluator simply performs arithmetic on the DP labels, with privacy preservation following from the post-processing property of DP. For circuits with only linear operations, the garbling overhead is essentially zero beyond the input encoding.

Evaluator-Half-Gates. Our arithmetic circuit supports multiplication $a \cdot b$ where the curator holds a secret value $a \in \mathbb{Z}_q$ and the platform holds a secret value $b \in \mathbb{Z}_p$. To securely compute this structured multiplication gate, we employ a mixed encoding: the curator's wire is encoded using a DP label obtained by adding calibrated geometric noise, while the platform's wire retains standard cryptographic labels assigned as random integers. The curator constructs garbled gate values that allow the platform to evaluate the multiplication using its input label without learning the curator's true value. The resulting output wire is itself a DP label, ensuring that privacy loss composes correctly across gates. Each evaluator-half-gate requires sending p integers, where p is the domain size of the platform's input b. For small domain sizes (e.g., $p = 100$ in our workload), this reduces communication by 87% compared to BMR's garbling [8]

Overall, our technique achieves 87% - 99% reductions in both communication and computational costs for standard 64-bit arithmetic operations, compared to state-of-the-art arithmetic garbling schemes. Complete construction details and proofs appear in Appendix C.

2.2 DP-S2PC Protocol for Query Processing

We now show how geometric garbling combines with oblivious transfer to construct a complete DP-S2PC protocol for the matrix mechanism. The data curator holds vectorized counts $\mathbf{x} \in \mathbb{N}^n$, and the platform holds a strategy matrix $\mathbf{S} \in \mathbb{Z}^{m \times n}$ designed to minimize the error of answers to the input query matrix $\mathbf{W}$ [35, 38]. The two parties jointly compute noisy measurements of $\mathbf{Sx}$ through the geometric mechanism [24], with the platform receiving the DP measurements.

The protocol achieves asymmetric privacy requirements through geometric garbling with sender-random oblivious transfer [4]: geometric garbling ensures ϵ-DP for $\mathbf{x}$, while oblivious transfer provides cryptographic

protection for $\mathbf{S}$. The privacy budget $\epsilon = \epsilon_{\text{in}} + \epsilon_{\text{gate}} + \epsilon_{\text{out}}$ is allocated across input encoding, evaluator-half-gates, and output noise. The protocol follows an offline–online structure. In the preprocessing (offline) phase, the parties execute oblivious transfer and generate the garbled circuit once. In the online phase, the curator provides the garbled inputs and the platform evaluates the circuit, requiring communication independent of the circuit size. Complete protocol specification appears in Appendix E.

Boosting Accuracy. Although answers to the input workload $\mathbf{W}$ can be obtained from the ϵ_{out}-DP measurements by solving a least-squares problem, this yields accuracy of $O(\epsilon_{out})$, where $\epsilon_{out} < \epsilon$. Instead, we improve accuracy to $O(\epsilon)$ via inverse-variance weighting [30]. Our key insight is to leverage all DP values obtained by the platform during protocol execution as computational artifacts, thereby utilizing the full privacy budget rather than discarding intermediate noisy measurements.

3 Experimental Results

We empirically evaluate the efficiency and accuracy of our DP-S2PC protocol, comparing it to various competing techniques, including the state-of-the-art arithmetic garbling technique (BMR) [8] and a non-private baseline in which the curator sends the raw data. We focus on the computational and communication costs, as well as the accuracy of the final query answers. We use five datasets from the DPBench benchmark [27] with domain sizes ranging from 128 to 4096. We evaluate prefix queries and all-range queries. All techniques are implemented in Rust, and all experiments were conducted on a MacBook Air (M2 chip with 16GB RAM).

Our technique significantly outperforms state-of-the-art BMR garbling in terms of runtime and network traffic. For instance, with a domain size of 1024, BMR requires 10 minutes and 488 MB of network traffic to execute the 2PC protocol. In contrast, our protocol completes in 3 minutes and consumes only 32MB of network traffic—an improvement of approximately 3× in runtime and 15× in network traffic. These improvements stem from free linear operations, compact DP wire labels, and optimized evaluator-half-gates.

In addition, our protocol incurs minimal online communication overhead. Any computation that can be performed independently of the dataset is considered offline pre-processing. During the online phase, our method requires minimal network traffic (e.g., 2KB for a domain size of 1024), matching the non-private baseline and thus achieving optimal efficiency.

Furthermore, we evaluate the root mean squared error (RMSE) of DP query answers and show that our approach achieves accuracy comparable to the non-private baseline by carefully allocating the privacy budget. For example, the RMSE of prefix queries was 6.13 under our protocol versus 6.03 for the optimal baseline. In contrast, applying DP locally on the curator's data resulted in RMSE that was 80% higher than the baseline.

Experimental setup details and extended results appear in Appendix F.

4 Conclusion and Future Work

We introduced DP garbling schemes that replace the cryptographic security of standard garbling with DP guarantees when evaluating DP mechanisms. Our geometric garbling construction demonstrates the practical potential of this approach for matrix multiplication, achieving a 3× speedup and a 15× reduction in communication compared to state-of-the-art arithmetic garbled circuits, while maintaining high accuracy. Our techniques enable privacy-preserving analytics platforms that allow organizations to securely outsource DP statistics publication by leveraging external expertise in mechanism design and optimization, while simultaneously protecting both individual privacy in the underlying database and the platform's proprietary optimization strategies.

There are several other exciting future directions. This involves adding the verifiability of computations for handling malicious settings, developing garbling techniques for general circuits, and extending the approach to a two-sided DP setting where a database is distributed across two parties. Additionally, an open question remains regarding the feasibility of replacing cryptographic wire labels in general garbled circuits with DP wire labels while improving efficiency.

References

[1] John M. Abowd. The u.s. census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 2867, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Accountability Act et al. Health insurance portability and accountability act of 1996. *Public law*, 104(191):1–16, 1996.

[3] Apple. Differential privacy, accessed: Sep 2022.

[4] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions. *Journal of Cryptology*, 30:805–858, 2017.

[5] Victor Balcer and Salil Vadhan. Differential privacy on finite computers. *arXiv preprint arXiv:1709.05396*, 2017.

[6] Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. Garbled neural networks are practical. *IACR Cryptol. ePrint Arch.*, 2019:338, 2019.

[7] Marshall Ball, Hanjun Li, Huijia Lin, and Tianren Liu. New ways to garble arithmetic circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–34. Springer, 2023.

[8] Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 565–577, New York, NY, USA, 2016. Association for Computing Machinery.

[9] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie Rogers. Shrinkwrap: efficient sql query processing in differentially private data federations. *Proceedings of the VLDB Endowment*, 12(3), 2018.

[10] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 479–488, 1996.

[11] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 451–468, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[12] Amos Beimel, Kobbi Nissim, and Mohammad Zaheri. Exploring differential obliviousness. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 65:1–65:20.

[13] Amos Beimel, Kobbi Nissim, and Mohammad Zaheri. Exploring differential obliviousness. *CoRR*, abs/1905.01373, 2019.

[14] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *Proceedings of the 18th International Conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'12, page 134–153, Berlin, Heidelberg, 2012. Springer-Verlag.

[15] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012.

[16] T.-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2448–2467. SIAM, 2019.

[17] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *International Conference on Cryptology and Information Security in Latin America*, pages 40–58. Springer, 2015.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, page 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[19] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[20] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[21] European Parliament and Council of the European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation). Official Journal of the European Union, 2016.

[22] Alireza Farhadi, MohammadTaghi Hajiaghayi, and Elaine Shi. Differentially private densest subgraph. *CoRR*, abs/2106.00508, 2021.

[23] Galois, Inc. swanky: A suite of rust libraries for secure computation. `https://github.com/GaloisInc/swanky`, 2019.

[24] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 351–360, 2009.

[25] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110):1–108, 1998.

[26] Adam Groce, Peter Rindal, and Mike Rosulek. Cheaper private set intersection via differentially private leakage. *Proceedings on Privacy Enhancing Technologies*, 2019(3):6–25, 2019.

[27] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 139–154, New York, NY, USA, 2016. Association for Computing Machinery.

[28] Xi He, Ashwin Machanavajjhala, Cheryl Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1389–1406, New York, NY, USA, 2017. Association for Computing Machinery.

[29] David Heath. Efficient arithmetic in garbled circuits. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 3–31. Springer, 2024.

[30] Julian PT Higgins, Sally Green, et al. Cochrane handbook for systematic reviews of interventions. 2008.

[31] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I 41*, pages 650–662. Springer, 2014.

[32] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Accessing data while preserving privacy. *CoRR*, abs/1706.01552, 2017.

[33] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II 35*, pages 486–498. Springer, 2008.

[34] Ilan Komargodski and Elaine Shi. Differentially oblivious turing machines. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*.

[35] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.

[36] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI global, 2005.

[37] Sahar Mazloom and S. Dov Gordon. Differentially private access patterns in secure computation. *IACR Cryptol. ePrint Arch.*, 2017:1016, 2017.

[38] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, June 2018.

[39] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661, 2012.

[40] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Annual International Cryptology Conference*, pages 126–142. Springer, 2009.

[41] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[42] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254, 1999.

[43] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on {OT} extension. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 797–812, 2014.

[44] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[45] Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. Dp-cryptography: marrying differential privacy and cryptography in emerging applications. *Communications of the ACM*, 64(2):84–93, 2021.

[46] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, page 162–167, USA, 1986. IEEE Computer Society.

[47] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. Low rank mechanism for optimizing batch queries under differential privacy. *arXiv preprint arXiv:1212.2309*, 2012.

[48] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 220–250. Springer, 2015.

A Preliminaries

Notation. Let $\mathbb{Z}_p$ denote the ring of integers modulo p. We denote $a \leftarrow \mathbb{Z}_p$ as a random integer uniformly sampled from $\mathbb{Z}_p$. We use $[m]$ to denote $\{1, ..., m\}$. A vector is represented by a bold lowercase letter $\mathbf{x}$, with its i-th entry denoted as $\mathbf{x}_i$. Similarly, a matrix is represented by a bold capital letter $\mathbf{S}$; $\mathbf{S}i$ denotes the i-th row vector and $\mathbf{S}_{i,j}$ denotes the entry at the i-th row and j-th column. The matrix-vector product is denoted as $\mathbf{Sx}$, and the element-wise matrix product as $\mathbf{S} \circ \mathbf{X}$. Notations $\mathbf{x}^i$ and $\mathbf{X}^i$ represent the vector and matrix labeled with i.

A.1 Data and Queries

Data. We consider a database $\mathbf{x}$ as a collection of records about individuals from a universe $\mathcal{X}$ with a relational schema $R(A_1, ..., A_d)$. Each attribute A_i has a finite domain $dom(A_i)$ with n_i possible values. Thus, the data domain $\mathcal{X}$ can be expressed as $\mathcal{X} = \Pi_{i=1}^{d} dom(A_i)$ with a domain size $n = \Pi_{i=1}^{d} n_i$. As a convenient representation, we will often view such a database as a histogram $\mathbf{x} \in \mathbb{N}^n$ [19, 35]. It is a vector of counts of length n, with each entry $\mathbf{x}_i$ corresponding to a tuple $t \in \mathcal{X}$ and counting the population with records equal to t.

Queries. We consider answering multiple queries on $\mathbf{x}$ at once. A query function $f(\mathbf{x}, w)$ is defined by the query parameters $w \in \mathcal{Q}$, where $\mathcal{Q}$ denotes the space of the query parameters. Formally, the evaluation of queries is denoted as a function $f : \mathcal{X} \times \mathcal{Q} \to \mathcal{Y}$. The query parameters $w \in \mathcal{Q}$ are often considered as auxiliary input fixed to f and thus we may implicitly write $f(\mathbf{x})$.

As a practical example, we will focus on a class of linear counting queries that computes a linear combination of the counts in the data vector $\mathbf{x}$. Linear queries can express a variety of common aggregation queries such as histograms, marginals, range queries, and data cubes. A linear query is defined by a row vector of length n, denoted as $\mathbf{s} \in \mathbb{R}^n$. The answer to a linear query $\mathbf{s}$ on $\mathbf{x}$ is the dot product $\mathbf{sx}$. Thus, a set of m linear counting queries can be defined by a query matrix $\boldsymbol{S} \in \mathbb{R}^{m \times n}$, with each row vector $\mathbf{S}_i$ representing a single query. The answers to m linear queries $\mathbf{S}$ over $\mathbf{x}$ is the matrix-vector multiplication $\mathbf{Sx}$. Formally, the evaluation of linear counting queries are defined as $f : \mathbb{N}^n \times \mathbb{R}^{m \times n} \to \mathbb{R}^m$, where $f(\mathbf{x}, \mathbf{S}) = \mathbf{Sx}$.

A.2 Differential Privacy

Differential Privacy (DP) ensures that the presence or absence of any single data entry has little effect on the output of a randomized mechanism $\mathcal{M}$. We denote two neighboring databases $\mathbf{x}, \mathbf{x}'$ that differ in at most one record as $\mathbf{x} \sim \mathbf{x}'$.

Definition A.1. ((ϵ, δ)-Differential Privacy [18])
A randomized mechanism $\mathcal{M} : \mathcal{X} \times \mathcal{Q} \to \mathcal{R}$ is (ϵ, δ)-DP if for all neighboring databases $\mathbf{x} \sim \mathbf{x}' \in \mathcal{X}$, all $w \in \mathcal{Q}$ and all $S \subseteq \mathcal{R}$,

$$\Pr[\mathcal{M}(\mathbf{x}, w) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathbf{x}', w) \in S] + \delta \, .$$

The parameter ϵ is called the privacy loss budget. When $\delta = 0$, it is pure DP, denoted as ϵ-DP. DP answers are typically obtained with a noise-addition mechanism, such as the Laplace mechanism [19]. We consider the discrete analogue of the Laplace mechanism, aka. the Geometric mechanism, which adds noise sampled from the two-sided geometric distribution, denoted as $\text{Geo}(\sigma)$, with p.d.f (of integer random variable z): $(p/2 - p)(1 - p)^{-|z|}$, where $p = e^{-1/\sigma}$. Let $Geo(\sigma)^m$ denote a vector of m independent samples from the distribution. The Geometric mechanism, defined below, satisfies ϵ-DP. We may assume that noise from $\text{Geo}(\cdot)$ is bounded e.g., in the context of a 64-bit integer representation [5].

Definition A.2. (Geometric Mechanism [24]) Given any function $f : \mathcal{X} \to \mathbb{R}^m$, the Geometric mechanism is defined as: $\text{GM}_{f,\epsilon}(\boldsymbol{x}) = f(\mathbf{x}) + \text{Geo}(\Delta_f/\epsilon)^m$.

The scale of the noise $\sigma = \Delta/\epsilon$ is determined by the privacy loss budget ϵ and the query sensitivity Δ_f, defined as follows. Specifically, the sensitivity of a set of queries defined by the matrix $\mathbf{S}$ is defined by the L1 norm of $\mathbf{S}$, denoted as $||\mathbf{S}||_1$. Thus, the Geometric mechanism for m by n query matrix $\mathbf{S}$ is computed as: $\mathbf{Sx} + \text{Geo}(||\mathbf{S}||_1/\epsilon)^m$.

Definition A.3. (L1-sensitivity [18]) The sensitivity of a query function $f : \mathcal{X} \to \mathbb{R}^m$ is: $\Delta_f = \max_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}} ||f(\mathbf{x}) - f(\mathbf{x}')||_1$.

We introduce some properties of DP mechanisms: the post-processing property states that all subsequent analyses on outputs produced by a ϵ-DP mechanism satisfy ϵ-DP, without degrading privacy. Additionally, the sequential composition property implies that if we answer two query sets under ϵ_1 and ϵ_2, respectively, the combined answers satisfy $(\epsilon_1 + \epsilon_2)$-DP.

Matrix Mechanism. Matrix mechanism [35] is a variant of noise-addition mechanisms designed for answering input queries defined by matrix $\mathbf{W}$ by using a set of underlying queries $\mathbf{S}$ such that there exists some $\mathbf{X}$ such that $\mathbf{W} = \mathbf{XS}$. $\mathbf{S}$ is referred to as the strategy matrix. The matrix mechanism noisily measures the strategy matrix $\mathbf{S}$ using the Geometric mechanism, i.e., $\tilde{\mathbf{y}} = \mathrm{GM}_\epsilon(\mathbf{x}, \mathbf{S})$. The answers to the input queries $\mathbf{W}$ can then be reconstructed by computing $\mathbf{WS}^+\tilde{\mathbf{y}}$. Here, $\mathbf{S}^+$ is a pseudo-inverse of $\mathbf{S}$. The matrix mechanism satisfies ϵ-DP. The state-of-the-art approach identifies a strategy by minimizing the mean-squared error (defined as follows) of the query answers [38].

Definition A.4 (Query error under strategy [35])**.** Given query vector $\mathbf{q}$ and strategy matrix $\mathbf{S}$, the error of the query answer using the matrix mechanism is defined as the expected squared error, which has the following closed-form expression: $\mathrm{Err}_\epsilon(\mathbf{q}, \mathbf{S}) = \mathrm{Var}_\epsilon(b_{\mathbf{S}})||\mathbf{qS}^+||_F^2$. Here, $\mathrm{Var}_\epsilon(b_{\mathbf{S}})$ denotes the variance of a i.i.d random variable of $b_{\mathbf{S}} \leftarrow \mathrm{Geo}(||\mathbf{S}||_1/\epsilon)$.

A.3 Garbled Circuits

Garbled circuits [46] are a core solution for secure two-party computation (S2PC). Roughly, the garbler transforms a circuit computing a function f into a "garbled" form that replaces gates with encrypted wire labels. The evaluator obliviously evaluates the garbled circuit to compute the function's output without learning any intermediate values.

Oblivious Transfer. Oblivious transfer (OT), a cryptographic primitive crucial in S2PC, involves a sender with paired messages (x_0, x_1) and a receiver with a bit $b \in \{0, 1\}$. By the end of the protocol, the sender learns nothing while the receiver learns x_b. This 1-out-of-2 OT extends to 1-out-of-n OT. In the context of Problem 1, the curator (as the sender) holds garbled input labels corresponding to all possible values of the platform's private input w. The platform (as the receiver) wishes to obtain the garbled input labels corresponding to his actual input, without revealing w to the curator. OT allows the platform to achieve this goal securely.

Numerous efficient OT protocols have been proposed [17, 42]. Specifically, the OT extension allow for millions of OTs to be performed per second [4, 10]. In this work, we utilize the sender random OT extension, a variant in which the sender's input can be random, thus reducing the communication from sender to receiver [4].

Definition A.5 ($t \times \mathrm{ROT}_q^p[4]$)**.** The sender (the curator) has no input and the receiver (the platform) has a vector of indices $\mathbf{c} = (c^1, \ldots, c^t) \in \mathbb{Z}_p^t$. After an execution of $t \times \mathrm{ROT}_q^p(\perp; \mathbf{c})$, the sender obtains t random vectors of $\{(k_0^i, \ldots, k_{p\text{-}1}^i)\}_{i\in[t]}$ for all $i, j : k_j^i \in \{0, 1\}^q$ and the receiver obtains a vector of t q-bit strings $(z^1, \ldots, z^t)$ such that for all $i \in [t]$, $z^i = k_{c^i}^i$.

B DP Garbling Scheme

We introduce an abstract definition of a DP garbling scheme for a randomized mechanism $\mathcal{M} : \mathcal{X} \times \mathcal{Q} \to \mathcal{R}$. We naturally extend the definition of DP to a garbling scheme framework for its privacy property by quantifying the total privacy leakage about $\mathbf{x} \in \mathcal{X}$ to an adversary, including from the circuit output itself. We emphasize that the privacy is defined with respect to the data input $\mathbf{x} \in \mathcal{X}$. Thus, we consider query parameters $w \in \mathcal{Q}$ as auxiliary input, distinguished from $\mathbf{x}$.

Definition B.1 (DP Garbling Scheme)**.** A ϵ-DP garbling scheme for a randomized mechanism $\mathcal{M} : \mathcal{X} \times \mathcal{Q} \to \mathcal{R}$, where $\mathcal{X}$ is the database space, $\mathcal{Q}$ is the parameter space (e.g., query strategies that defines the mechanism), and $\mathcal{R}$ is the output space, is composed of the following algorithms.

- **Gb:** On the security parameter λ, privacy parameter ϵ, and an a randomized mechanism $\mathcal{M}$, outputs (F, e, e_w, d), where F is a garbled circuit, and e, e_w is encoding information for database input and parameter input, and d is decoding information.
- **En:** On input $(e, \mathbf{x})$, where e is as above and $\mathbf{x} \in \mathcal{X}$ is a database input, outputs a garbled input X.

- **EnAux:** On input (e_w, w), where e_w is as above and $w \in \mathcal{Q}$ is a parameter input, outputs a garbled input W.
- **Ev:** On input (F, X, W) as above, outputs a garbled output Y.
- **De:** On input (d, Y) as above, outputs $\tilde{y} \in \mathcal{R}$.

The correctness property is: if $(F, e, e_w, d) \leftarrow Gb(\lambda, \epsilon, \mathcal{M})$, for all $\mathbf{x} \in \mathcal{X}$ and $w \in \mathcal{Q}$, the output of $De(d, Ev(F, En(e, \mathbf{x}), EnAux(e_w, w)))$ and $\mathcal{M}(\mathbf{x}, w)$ are identically distributed.

We require the following privacy properties with respect to $\mathbf{x}$, where we assume a computationally bounded semi-honest adversary.

- **Privacy:** (F, X, d) should reveal no information about $\mathbf{x}$ beyond what can be captured by the definition of ϵ-DP.

  Definition B.2 (ϵ-privacy)**.** Consider an experiment $b \in \{0, 1\}$ that adversary $\mathcal{A}$ outputs two neighboring datasets $\mathbf{x}_0 \sim \mathbf{x}_1 \in \mathcal{X}$, auxiliary input w and a randomized mechanism $\mathcal{M}$ and receives a tuple (F, X, Q, d) where $(F, e, e_w, d) \leftarrow Gb(\lambda, \epsilon, \mathcal{M})$, $X \leftarrow En(e, \mathbf{x})$ and $Q \leftarrow EnAux(e_w, w)$. Let $\text{VIEW}^{pri}_{\mathcal{A}}(\mathbf{x}_b)$ be the view of adversary $\mathcal{A}$ in the experiment b, including the mechanism output. We say that a DP garbling scheme satisfies ϵ-privacy if for all possible sets $\mathcal{V}_{\mathcal{A}}$ of views of $\mathcal{A}$, it holds that:

$$\Pr[\text{VIEW}^{pri}_{\mathcal{A}}(\mathbf{x}_0) \in \mathcal{V}_{\mathcal{A}}] \leq e^{\epsilon} \Pr[\text{VIEW}^{pri}_{\mathcal{A}}(\mathbf{x}_1) \in \mathcal{V}_{\mathcal{A}}] + \text{negl}(\lambda).$$

- **Obliviousness:**(F, X) (without d) should reveal no information about $\mathbf{x}$ beyond what can be captured by the definition of ϵ-DP.

  Definition B.3 (ϵ-obliviousness)**.** Consider an experiment $b \in \{0, 1\}$ that adversary $\mathcal{A}$ outputs two neighboring datasets $\mathbf{x}_0 \sim \mathbf{x}_1 \in \mathcal{X}$, auxiliary input w and a randomized mechanism $\mathcal{M}$ and receives a tuple (F, X, Q) where $(F, e, e_w, d) \leftarrow Gb(\lambda, \epsilon, \mathcal{M})$, $X \leftarrow En(e, \mathbf{x})$ and $Q \leftarrow EnAux(e_w, w)$. Let $\text{VIEW}^{obl}_{\mathcal{A}}(x_b)$ be the view of adversary $\mathcal{A}$ in the experiment b. We say that a DP garbling scheme satisfies ϵ-obliviousness if for all possible sets $\mathcal{V}_{\mathcal{A}}$ of views of $\mathcal{A}$, it holds that:

$$\Pr[\text{VIEW}^{obl}_{\mathcal{A}}(\mathbf{x}_0) \in \mathcal{V}_{\mathcal{A}}] \leq e^{\epsilon} \Pr[\text{VIEW}^{obl}_{\mathcal{A}}(\mathbf{x}_1) \in \mathcal{V}_{\mathcal{A}}] + \text{negl}(\lambda).$$

Above, we provide the computational DP [40] to the obliviousness and privacy properties. $\text{negl}(\lambda)$ refers to any function that is $o(\lambda^{-c})$ for all constants c. $\text{VIEW}_{\mathcal{A}}(\mathbf{x})$ denotes the view of adversary during the experiment. It consists of a tuple of $(r, F, X, Q, d, \tilde{y}, t_1, \ldots, t_l)$, where r is $\mathcal{A}$'s private randomness, F, X, Q, d are as above (d is excluded for the obliviousness property), $\tilde{y}$ is the circuit output and $t_1, \ldots, t_l$ are messages that $\mathcal{A}$ observes during the circuit evaluation. We ask that this view be DP with respect to $\mathbf{x}$.

The definitions above imply a DP garbling scheme that satisfies ϵ-privacy holds ϵ-obliviousness. In addition, ϵ-privacy immediately implies that a mechanism $\mathcal{M}$ to be computed must be ϵ_{out}-DP for some $\epsilon_{out} \leq \epsilon$. In the context of information-theoretic-DP, we will remove the security parameter from the above definitions. In fact, we will construct an information-theoretic ϵ-DP garbling scheme that is secure against even a computationally unbounded adversary.

Privacy of auxiliary input. The privacy of a DP garbling scheme is defined with respect to the privacy of $\mathbf{x}$, reflecting the original definition of DP. Thus, the DP garbling scheme defined above does not consider the privacy of auxiliary input w. This input is known to the adversary, where it can be public or secret known to the adversarial evaluator. This assumption naturally reflects DP-S2PC.

We highlight the major modifications to the standard garbling scheme: 1) a DP garbling scheme must compute a DP mechanism. 2) The privacy of a DP garbling scheme is captured by the definition of DP, which quantifies the privacy leakage even from the mechanism output itself.

procedure $\textsc{Gb}(\epsilon, \mathrm{GM}_{f,\epsilon_{out}})$
  Allocate $\epsilon_{in}, \epsilon_g$ st. $\epsilon_{in} + h\epsilon_g + \epsilon_{out} = \epsilon$
  $L_i \leftarrow \texttt{Geo}(1/\epsilon_{in}), \forall i \in \texttt{Input}$
  $\mathbf{k}^i = (\hat{L}_i^0, \ldots, \hat{L}_i^{p_i-1}) \leftarrow \mathbb{Z}_q^{|i.\texttt{domain}|}, \forall i \in \texttt{AuxMul}$
  $e \leftarrow \{L_i\}_{i\in\texttt{Input}}, e_w \leftarrow \{\mathbf{k}^i\}_{i\in\texttt{AuxMul}}$
  $L_i \leftarrow 0, \forall i \in \texttt{AuxAdd}$, $L_i \leftarrow c_i, \forall i \in \texttt{Constant}$
  for $g \in f.\texttt{gates}_{\texttt{topo}}$ **do**
    $i, j \leftarrow g.\texttt{input}, k \leftarrow g.\texttt{output}$
    if g is add, eval-add, c-mul **then**
      $L_k \leftarrow g(L_i, L_j)$
    else if g is $\mathbb{Z}_{p_i}$-eval-half **then**
      $L_k \leftarrow \texttt{Geo}(\Delta_g/\epsilon_g)$
      $G_k^b \leftarrow b \cdot L_i + \hat{L}_j^b - L_k, \forall b = 0 \ldots p_i - 1$
      $G_k \leftarrow \{G_k^0, \ldots, G_k^{p_i-1}\}$
    end if
  end for
  $d_i \leftarrow L_i - \texttt{Geo}(\Delta_f/\epsilon_{out}), \forall i \in \texttt{Output}$
  return $F \leftarrow \{G_k\}_{k\in f.\texttt{eval-half}}, e, e_w, d$
end procedure

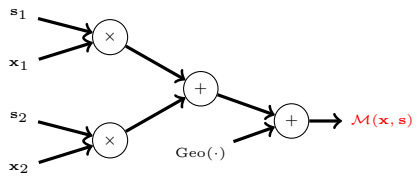
procedure $\textsc{En}(e, \mathbf{x})$
  $X_i \leftarrow \mathbf{x}_i + e_i, \forall i \in \texttt{Input}$
  return $X \leftarrow \{X_i\}_{i\in\texttt{Input}}$
end procedure

procedure $\textsc{EnAux}(e_w, w)$
  Parse e_w as $\{\mathbf{k}^i\}_{i\in\texttt{AuxMul}}$
  for $i \in \texttt{AuxMul}$ **do**
    Parse $\mathbf{k}^i$ as $(k_i^0, \ldots, k_i^{|i.\texttt{domain}|-1})$
    $Q_i \leftarrow k_i^{w_i}$
  end for
  return $\{Q_i\}_{i\in\texttt{AuxMul}}$
end procedure

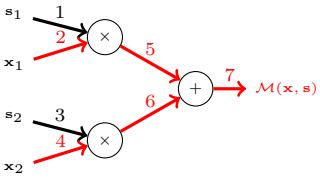
procedure $\textsc{De}(d, Y)$
  $\tilde{y}_i \leftarrow Y_i - d_i, \forall i \in \texttt{Output}$
  return $\tilde{y} \leftarrow \{\tilde{y}_i\}_{i\in\texttt{Output}}$
end procedure

procedure $\textsc{Ev}(F, X, Q)$
  $L_i \leftarrow X_i, \forall i \in \texttt{Input}$
  $\hat{L}_i \leftarrow Q_i, \forall i \in \texttt{AuxMul}$
  $L_i \leftarrow b_i, \forall i \in \texttt{AuxAdd}$
  $L_i \leftarrow c_i, \forall i \in \texttt{Constant}$
  for $g \in f.\texttt{gates}_{\texttt{topo}}$ **do**
    $i, j \leftarrow g.\texttt{input}, k \leftarrow g.\texttt{output}$
    if g is add, eval-add, c-mul **then**
      $L_k \leftarrow g(L_i, L_j)$
    else if g is b-eval-half **then**
      $L_k \leftarrow b \cdot L_i + \hat{L}_j - G_k^b$
    end if
  end for
  $Y_i \leftarrow L_i, \forall i \in \texttt{Output}$
  return $Y \leftarrow \{Y_i\}_{i\in\texttt{Output}}$
end procedure

Figure 1: Geometric garbling: ϵ-DP Garbling scheme for the Geometric mechanism $\mathrm{GM}_f(\mathbf{x}, w)$. The function f is represented as an arithmetic circuit over $\mathbb{Z}_q$ as defined in Section C.1.We denote h as the total number of evaluator-half-gates in the circuit.



(a) Standard garbled circuit. The ϵ-privacy budget is concentrated on mechanism output.



(b) DP garbled circuit. The ϵ-privacy budget is distributed across wires as well as mechanism output.

Figure 2: Evaluating a DP mechanism $\mathcal{M}(\mathbf{x}, \mathbf{s})$ for dot product in Example 1 under the privacy budget of ϵ. The wires in red bound the privacy loss through DP. The wires in black reveal no information.

C Geometric Garbling Scheme

We introduce a DP garbling scheme tailored for computing the Geometric mechanism GM_f for a function $f(\mathbf{x}, w)$, which we call *geometric garbling scheme*. While our techniques are not as general as standard garbled circuits, they are specifically designed to support matrix multiplication within 2PC. In this setup, $\mathbf{x}$ is secret inputs known to the garbler while w is secret inputs known to the evaluator.

C.1 Arithmetic Circuit for Matrix Multiplication

We view a function $f(\mathbf{x}, w) : \mathbb{F}^n \times \mathbb{F}^l \to \mathbb{F}^m$ as an arithmetic circuit composed of gates and wires. There are n data input wires (`Input`), l auxiliary input wires (`Aux`) and m output wires (`Output`). While `Input` serves for the garbler's secret inputs $\mathbf{x}$, `Aux` serves for the evaluator's secret inputs w. Within `Aux`, we distinguish input wires for addition (`AuxAdd`) and those for multiplication (`AuxMul`). Public constants can be hard-coded into the circuit (`Constant`).

In the circuit, each wire is associated with an index i and each gate is associated with an index of its output wire. The functionality of a gate is determined by a fan-in-two arithmetic gate $g : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$, where $g(a, b) = c$ denotes that the gate takes two input wires carrying values $a, b \in \mathbb{F}$ and outputs a wire carrying a value $c \in \mathbb{F}$. The field $\mathbb{F}$ is typically considered to be integers modulo q, $\mathbb{Z}_q$.

We focus on the following types of gates to support matrix multiplication in 2PC: 1) addition (*add*) 2) addition where one of the secret inputs is known to the evaluator (*eval-add*), 3) multiplication-by-constant (*c-mul*), and 4) multiplication where one of the secret inputs is known to the evaluator (*eval-half*). Below,

we provide example circuits for basic vector operations. Additionally, our circuit can support multilevel matrix multiplication as in Algorithm 3.

We note that the topology of the circuit, number of wires, number of gates, and the type of gates are considered public.

Example 1 (Arithmetic circuit of dot product)**.** *Figure 2b shows an arithmetic circuit* $f(\mathbf{x}, \mathbf{s})$ *which computes* $\mathbf{s}_1\mathbf{x}_1 + \mathbf{s}_2\mathbf{x}_2$*. It has two input wires, i.e.,* $\texttt{Input} = \{2, 4\}$*, two auxiliary input wires, i.e.,* $\texttt{AuxMul} = \{1, 3\}$*, and a single output wire,* $\texttt{Output} = \{7\}$*. There are two evaluator-half-gates* g_5, g_6 *and a single addition gate* g_7*.*

C.2 DP Wire Label Representation

We introduce an efficient wire label representation within the circuit. Every wire except for `Aux` is encoded into a wire label to ensure that the privacy leakage about $\mathbf{x}$ from each wire is bounded under DP, which we refer to as a *DP wire label*. This is achieved by applying a DP mechanism to each gate in the circuit. The wires on `Aux` are treated differently depending on the gate types, as we will explain in the following sections.

Formally, consider a gate g that computes $g(a, b) = c$. We apply the Geometric mechanism to the gate function g to encode an output wire carrying an integer $c \in \mathbb{Z}_q$ into a wire label, denoted as L^c. We may write L^c_k to explicitly denote the wire indexed with k. Every wire label (except for `Aux`) in the circuit has a form $L^c_k = c + r_k$, where r_k is DP-noise (random integer) unique to the wire k. This noise term r_k is independent of the actual input and is assigned to the corresponding wire as a *zero-label* during a garbling phase, denoted as L^0_k.

Definition C.1 (Geometric Gate Mechanism)**.** Let g be a gate within a circuit $f(\mathbf{x})$ that computes $h_g(\mathbf{x})$. The following Geometric gate mechanism $\mathcal{G}_g$ for gate g satisfies ϵ-DP: $\mathcal{G}_g(\mathbf{x}) = h_g(\mathbf{x}) + \mathrm{Geo}(\Delta_g/\epsilon)$.

Above definition is essentially equivalent to Definition A.2. Here, Δ_g denotes the sensitivity of the gate function g, i.e., $\Delta_g = \max_{\mathbf{x} \sim \mathbf{x}' \in \mathcal{X}} ||h_g(\mathbf{x}) - h_g(\mathbf{x}')||_1$. We assume the gate sensitivity Δ_g is a public value as it typically depends on query parameters w which may be secret to the garbled-circuit generator.

Input wire labels for `Input` are a special case of Definition C.1 with a unary identity gate (i.e., the output of the gate is its input). Thus, each input wire label carrying $\mathbf{x}_i \in \mathbf{x}$ is encoded as $\mathbf{x}_i + \mathrm{Geo}(1/\epsilon)$.

Length of Wire Labels. Our wire label representation is efficient, requiring at most $\log q$ bits, where q is the largest integer supported during the computation. For example, with a 64-bit arithmetic circuit, our technique requires at most 64 bits for each wire label while BMR garbling requires 2048 bits. Furthermore, for each input wire label carrying an integer $\mathbf{x}_i$, the corresponding communication cost provides a tighter bound of $O(\log \mathbf{x}_i + \log 1/\epsilon)$ with minimal overhead, particularly when ϵ is large.

C.3 DP Linear Gates

We show that addition, evaluator-addition and multiplication-by-constant gates can be evaluated for free without incurring additional cost.

Consider an arithmetic gate with two input wires, denoted as i and j, and one output wire denoted as k, represented as $g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$. At least one of the input wires and the output wire are encoded as DP wire labels. Recall that every DP wire label has a form $L^a_i = a + r_i$ for some random DP-noise value $r_i \in \mathbb{Z}$. During the garbling phase, we assign the random value (represented below as underlined) as the zero-label of the output wire k. This allows the evaluator to simply evaluate the gate locally during the evaluation phase.

- **Add-gate.** For an addition gate of $g(a, b) = a + b$, given two input DP wire labels L^a_i and L^b_j, the output wire label for $a + b$ is obtained for free since:

$$L^a_i + L^b_j = (a + r_i) + (b + r_j) = (a + b) + \underline{(r_i + r_j)} = L^{a+b}_k.$$

  This can be naturally extended to a fan-in n addition gate.

- **Evaluator-add-gate.** Above, when the secret value b is known to the evaluator, with the received DP wire label L_i^a, the evaluator can perform the addition gate for free since:

$$L_i^a + b = (a + b) + \underline{r_i} = L_k^{a+b}.$$

  Notice that a wire label for the evaluator's input is not required.

- **c-Mul-gate.** For a multiplication-by-a-constant gate $g(a, c) = ac$ (public constant c), given an input DP wire label of L_i^a, the output wire label for ac is obtained for free since:

$$c \cdot L_i^a = c(a + r_i) = ac + \underline{cr_i} = L_k^{ac}.$$

The evaluation of DP input wire labels produces the corresponding DP output wire label of an output value, similar to the concept of free-XOR [33, 8] that can be evaluated at no cost. In case when a circuit is only composed of linear gates, the resulting garbled circuit can achieve high efficiency since there is no cost of garbling. Additionally, evaluating the above DP linear gates do not degrade privacy due to the post-processing theorem.

C.4 DP Evaluator-Half-Gates

We show how to garble a multiplication $a \cdot b$ when the secret value b is known to the evaluator, referred to as the *DP evaluator-half-gate*.

Consider a fan-in-two multiplication gate $g : \mathbb{Z}_q \times \mathbb{Z}_p \to \mathbb{Z}_q$, where $g(a_{\mathbf{x}}, b) = ab$. Here, $a_{\mathbf{x}}$ denotes its dependency on $\mathbf{x}$ in the circuit $f(\mathbf{x})$. In contrast, b is independent of $\mathbf{x}$ but a secret value known to the evaluator. The corresponding input wires are indexed with i and j and the output wire is indexed with k.

We first employ mix-wire-labels in our scheme. On any wire that carries a value dependent on $\mathbf{x}$ (e.g., wires i, k from above) will be encoded by DP wire labels, following Section C.2. We recall that a DP wire label encoding a has the form of $L_i^a = a + r_i$ where r_i is a random DP-noise. On the other hand, for a wire that is irrelevant to $\mathbf{x}$ (e.g., wire j from above), we employ "crypto wire labels" (as opposed to DP) by assigning a random label $\hat{L}_j^{b'} \leftarrow \mathbb{Z}_q$ for each possible value $b' \in \mathbb{Z}_p$ that the wire j can take. We use the hat notation for crypto wire labels to distinguish from DP wire labels.

During the garbling, for every $b' \in \mathbb{Z}_p$, the garbler uses $\hat{L}_j^{b'}$ as a one-time pad to encode the value $b' \cdot L_i^0 - L_k^0$ as follows, where L_i^0 and L_k^0 are the zero-labels of the input and the output wires:

$$G_k^{b'} = \hat{L}_j^{b'} + b' \cdot L_i^0 - L_k^0.$$

At evaluation time, the evaluator receives the p random integers $G_k^0, \ldots, G_k^{p-1}$, as well as the DP wire label $L_i^a = a + r_i$ and the crypto wire label $\hat{L}_j^b$ (a is unknown but b is known). The evaluator "opens" the appropriate encoding G_k^b as follows:

$$b \cdot L_i^a + \hat{L}_j^b - G_k^b.$$

This represents the DP wire label encoding ab on the output wire k since:

$$b \cdot L_i^a + \hat{L}_j^b - G_k^b = b(a + r_i) + (\hat{L}_j^b) - (\hat{L}_j^b + b \cdot r_i - r_k) = ab + r_k.$$

Every random integer $G_k^{b'}$ looks completely random to the evaluator since the label $\hat{L}_j^{b'}$ generated by the garbler serves as a one-time-pad. In addition, given input wire labels $L_i^a, \hat{L}_j^b$, the evaluator only learns the DP output wire label $L_k^{ab} = ab + r_k$ and nothing more, provided that r_i and r_k are kept secret by the garbler.

The garbled evaluator-half-gate consists of p random integers, each of which is represented by $\log q$ bits. Hence, the total garbling cost is $p \log q$ bits per gate. This is especially efficient when the domain size of b, i.e., $\mathbb{Z}_p$, is small (e.g., $p = 100$ in our application).

C.5 Decoding

We present our efficient decoding techniques. A key aspect of a DP garbling scheme is that the decoding algorithm must correctly output the desired DP mechanism output $\mathcal{M}_{f,\epsilon_{out}}(\mathbf{x})$. Unlike standard garbled circuits, we never release non-DP output $f(\mathbf{x})$.

Without loss of generality, let us consider a fan-in-two gate $g_k(a, b)$ over integers a and b. The output wire is encoded with a DP wire label $L_k^{g(a,b)} = g_k(a, b) + r_k$, where r_k is a random noise. Our goal is to decode this wire label to produce the desired output of the ϵ_{out}-DP Geometric mechanism: $g_k(a, b) + s_k$, where $s_k \leftarrow \text{Geo}(\Delta_k/\epsilon_{out})$.

Our solution is straightforward: we subtract the DP-noise term r_k from the DP wire label and add the desired DP-noise s_k to obtain the correct mechanism output. During the garbling phase, the garbler secretly generates a random integer $d_k = (r_k - s_k)$ as the decoding information. The random noise r_k and s_k must be kept secret by the garbler. In the evaluation phase, the evaluator receives the value d_k along with the DP wire label $L_k^{g(a,b)}$. The evaluator then computes $L_k^{g(a,b)} - d_k$ to derive the desired DP output:

$$L_k^{g(a,b)} - d_k = g_k(a, b) + r_k - (r_k - s_k) = g_k(a, b) + s_k.$$

Our decoding technique is efficient as each output wire only requires unique decoding information. Whereas, existing garbling techniques typically require q calls of cryptographic hash functions, where q is the domain size of the output wire.

C.6 Our Construction

Combining all the building blocks presented in the previous sections, we show our geometric garbling scheme in Figure 1. This technique supports an arithmetic circuit for matrix multiplication as described in Section C.1. While other noise-addition mechanisms like the Laplace or Gaussian mechanisms are applicable, floating-point implementations may result in the violation of DP [39].

Within our scheme, we divide the privacy loss budget ϵ into ϵ_{in}, ϵ_g, and ϵ_{out}, such that $\epsilon_{in} + h\epsilon_g + \epsilon_{out} = \epsilon$, where h denotes the number of evaluator-half-gates in the circuit. These parameters quantify potential privacy leakage about $\mathbf{x}$ during the evaluation of garbled circuits. Specifically, ϵ_{out} protects mechanism output, ϵ_{in} encodes input wires (`Input`), and ϵ_g encodes output wires of evaluator-half-gates. Notice the linear gates (i.e., add, eval-add, c-mul) are evaluated without consuming privacy budget, as they do not degrade privacy due to the post-processing theorem.

The privacy budget allocation in Figure 1 is general and may be overly conservative for certain circuits. For instance, in the dot product circuit in Example 1, we can apply sequential composition theorem to the multiplication gates since they are independent of each other. Section E provides a refined privacy budget allocation tailored to our specific application and describes how to configure the privacy loss parameters.

Cost Comparison to BMR Garbling. We compare the garbling cost of various operations between our technique and BMR garbling. As summarized in Table 1, our technique achieves substantial improvements of 87% to 99% in both communication and computational costs for our application of 64-bit arithmetic operations. The cost of evaluator-half-gates, proportional to the evaluator's input domain size p, may vary depending on applications; $p = 100$ in our application. Our efficiency attributes to an efficient wire label representation with constant overhead, unlike BMR garbling, which requires a wire label size proportional to the security parameter. Moreover, our techniques rely solely on arithmetic operations, eliminating the need for expensive cryptographic primitives such as hash functions.

Correctness Proof. We show the correctness of the geometric garbling in Figure 1. That is, we show that running $De(d, Ev(F, En(e, \mathbf{x}), EnAux(e_w, w)))$, where $(F, e, e_w, d) \leftarrow Gb(\epsilon, \text{GM}_{\epsilon_{out}})$, correctly outputs $\text{GM}_{\epsilon_{out}}(\mathbf{x}, w)$.

Proof. Within the circuit $f : \mathbb{Z}_q^n \times \mathbb{Z}_q^l \to \mathbb{Z}_q^m$ in Figure 1, let $f(\mathbf{x}, w; i) \in \mathbb{Z}_q$ denotes the i-th output of $f(\mathbf{x}, w; i)$. Without loss of generality, we show that each output wire $i \in [m]$ in the circuit correctly outputs

Table 1: Concrete cost comparison of garbling various operations on 64-bit arithmetic operations with $p = 100$, as exemplified by our application. The third column indicates communication costs, while the fourth and fifth columns denote computational costs for garbling and evaluation, in terms of the number of hash function calls (H) and the number of arithmetic operations (# ope.).

	Scheme	Size (bits)	Garb. cost		Eval. cost	
			H	# ope.	H	# ope.
Add	BMR	0	0	649	0	649
c-Mul	Ours	0	0	1	0	1
$\mathbb{Z}_p$-eval-half	BMR	48768	381	10651	16	649
	Ours	6400	0	100	0	1
Input wire label	BMR	2048	0	649	N/A	N/A
per integer	Ours	$1 \sim 64$	0	1	N/A	N/A
Decoding info.	BMR	48768	381	10651	381	381
per output value	Ours	$1 \sim 64$	0	1	0	1

$f(\mathbf{x}, w; i) + \mathrm{Geo}(\Delta_f/\epsilon_{out})$:

$$\begin{aligned}\tilde{y}_i &= Y_i - d_i \\ &= L_i^{f(\mathbf{x},w;i)} - (L_i^0 - \mathrm{Geo}(\Delta_f/\epsilon_{out})) \\ &= f(\mathbf{x}, w; i) + L_i^0 - L_i^0 + \mathrm{Geo}(\Delta_f/\epsilon_{out}) \\ &= f(\mathbf{x}, w; i) + \mathrm{Geo}(\Delta_f/\epsilon_{out}).\end{aligned}$$

This completes the proof by substitution and the property of DP wire labels. Above, $L_i^{f(\mathbf{x},w;i)}$ denotes a DP wire label of the output wire i that carries $f(\mathbf{x}, w; i)$ and L_i^0 is its zero-label. □

Privacy Proof. We show that the geometric garbling in Figure 1 satisfies $(\epsilon_{in} + h\epsilon_g + \epsilon_{out})$-privacy and $(\epsilon_{in} + h\epsilon_g)$-obliviousness.

Proof. We first show $(\epsilon_{in}+h\epsilon_g+\epsilon_{out})$-privacy. All we need to show is that the adversary's view is differentially private wrt. $\mathbf{x}$. The view of adversary, $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x})$, consists of a tuple $(F, X, Q, d, L_1, \ldots, L_t, \tilde{y})$, where $(F, e, e_w, d) \leftarrow Gb(\epsilon, \mathrm{GM}_{\epsilon_{out}}), X \leftarrow En(e, \mathbf{x}), Q \leftarrow EnAux(e_w, w), Y \leftarrow Ev(F, X, Q), \tilde{y} \leftarrow De(d, Y)$ in Figure 1. In addition, $L_1, \ldots, L_t$ denote all DP wire labels obtained during the evaluation of the garbled circuit F, assuming the total of t gates in the circuit. Given a tuple of (F, X, Q), the adversary can only open and learn the appropriate garbled output of each garbled multiplication gate. Furthermore, additionally knowing d will only reveal the outputs $\tilde{y}$. Note that all the DP-noise generated from the garbling algorithm must be kept secret.

We now consider $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x})$ as a randomized mechanism that outputs the above tuple of random variables. Our proof sketch is to show this mechanism is ϵ-DP, which shows the ϵ-privacy of Figure 1.

We first show that the garbled circuit F, the garbled query Q, and the decoding information d themselves do not reveal anything about $\mathbf{x}$. The garbled query Q does not reveal anything about $\mathbf{x}$ as they are uniformly random values independent of $\mathbf{x}$. In addition, the garbled circuit F is completely random; G_k^b in Line 13 is completely random since $\hat{L}_j^b$ sampled from $\mathbb{Z}_q^{p_i}$ can be considered as an one-time pad. Furthermore, d_i does not give additional information about $\mathbf{x}$ beyond what the adversary already knew from their view since $d = L_i - \tilde{y}_i$, where L_i is the DP wire label of the output gate i.

From the above results, it suffices to show that the $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x})$ that outputs a tuple of $(X, L_1, \ldots, L_t, \tilde{y})$ satisfies ϵ-DP. We show this by the sequential composition theorem. More specifically, we consider $\mathrm{VIEW}_{\mathcal{A}}$ as the composition of three mechanisms, denoted as $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x}) = (\mathcal{E}, \mathcal{G}_1, \ldots, \mathcal{G}_t, \mathcal{D})$, where $\mathcal{E}(\mathbf{x})$ is an Encoding mechanism that outputs the garbled input X, $\mathcal{G}_i(\mathbf{x})$ is an Gate evaluation mechanism of each gate $i \in [t]$ that outputs the garbled output L_i and $\mathcal{D}(\mathbf{x})$ is an Decoding mechanism that outputs $\tilde{y}$.

First, the encoding mechanism can be formulated as: $\mathcal{E}(\mathbf{x}) = \mathbf{x} + \mathrm{Geo}(1/\epsilon_{in})^n$. Thus, following the privacy of the Geometric mechanism, the mechanism $\mathcal{E}$ satisfies ϵ_{in}-DP.

Second, we formulate the gate evaluation mechanism of each gate. Importantly, we only need to consider evaluator-half-gates since the evaluation of linear gates does not degrade privacy due to the post-processing theorem. Following Definition C.1, our gate evaluation mechanism of the evaluator-half-gate g_i can be expressed as: $\mathcal{G}_{g_i}(\mathbf{x}) = h_g(\mathbf{x}) + \mathrm{Geo}(\Delta_g/\epsilon_g)$, where $h_g(\mathbf{x})$ is the gate output value. Thus, the mechanism $\mathcal{G}_{g_i}$ satisfies ϵ_g-DP by the definition.

Lastly, our decoding mechanism is $\mathcal{D}(\mathbf{x}) = \mathrm{GM}_{\epsilon_{out}}(\mathbf{x})$ itself. By the correctness definition of a DP garbling scheme, $\mathcal{D}$ satisfies ϵ_{out}-DP.

Hence, from the sequential composition theorem, the composite mechanism $\mathrm{VIEW}_{\mathcal{A}}$ satisfies ϵ-DP, where $\epsilon = \epsilon_{in} + h\epsilon_g + \epsilon_{out}$ and h denotes the total number of evaluator-half-gates in the circuit. Thus, the geometric garbling in Figure 1 satisfies ϵ-privacy.

For the obliviousness proof, we can apply the same proof as above except that we consider the adversary does not obtain the decoding information. Without the decoding information, the adversary will not learn the output $\tilde{y} = \mathcal{M}(\mathbf{x})$. This makes a small change to the above proof, where the view $\mathrm{VIEW}_{\mathcal{A}}$ is now composed of the two mechanisms: ϵ_{in}-DP encoding mechanism $\mathcal{E}$ and ϵ_g-DP gate evaluation mechanism $\mathcal{G}_i$. Hence, from the sequential composition theorem, $\mathrm{VIEW}_{\mathcal{A}}$ satisfies $(\epsilon_{in} + h\epsilon_g)$-DP. Thus, the geometric garbling in Figure 1 satisfies $(\epsilon_{in} + h\epsilon_g)$-obliviousness. □

D DP-S2PC Protocols with Geometric Garbling

We provide the formal privacy definition of DP-S2PC and present our online-efficient protocol using the geometric garbling scheme.

D.1 Privacy Definition

We naturally extend the definition of DP to a protocol between two parties a curator with private dataset $\mathbf{x}$ and a platform with (possibly private) query parameters w, by requiring that the platform's view be DP with respect to the curator's input $\mathbf{x}$. Following that, we define DP-S2PC where we modify the standard indistinguishability-based security definition of S2PC by providing the computational DP [40] to the curator's privacy. The platform's privacy remains the same.

The following definition assumes semi-honest parties and they are computationally bounded.

Definition D.1 (DP-S2PC)**.** A two-party protocol Π for computing a randomized mechanism $\mathcal{M} : \mathcal{X} \times \mathcal{Q} \to \mathcal{R}$ satisfies ϵ-DP-S2PC if for any two neighboring datasets $\mathbf{x} \sim \mathbf{x}' \in \mathcal{X}$, any query $w \in \mathcal{Q}$, and for all possible sets $\mathcal{V}_B$ of the platform's views, it holds that $\Pr[\mathrm{VIEW}_B^{\Pi}(\mathbf{x}, w) \in \mathcal{V}_B] \leq e^{\epsilon} \Pr[\mathrm{VIEW}_B^{\Pi}(\mathbf{x}', w) \in \mathcal{V}_B] + \mathrm{negl}(\lambda)$, and if for any query $w, w' \in \mathcal{Q}$, any dataset $\mathbf{x} \in \mathcal{X}$, and for all possible sets $\mathcal{V}_A$ of the curator's views, it holds that $\Pr[\mathrm{VIEW}_A^{\Pi}(\mathbf{x}, w) \in \mathcal{V}_A] \leq \Pr[\mathrm{VIEW}_A^{\Pi}(\mathbf{x}, w') \in \mathcal{V}_A] + \mathrm{negl}(\lambda)$.

Above, $\mathrm{VIEW}_A^{\Pi}(\mathbf{x}, \cdot)$ ($\mathrm{VIEW}_B^{\Pi}(\cdot, w)$ resp.) denotes the curator's view (platform resp.) during an execution of Π. Specifically, the platform's view, $\mathrm{VIEW}_B^{\Pi}(\cdot, w)$, includes w, the platform's private randomness, messages that the platform has received during the protocol execution. The output received by the platform at the end of the protocol is also implicitly included in the view and it should be distributed identically with $\mathcal{M}(\mathbf{x}, w)$. The curator's view, $\mathrm{VIEW}_A^{\Pi}(\mathbf{x}, \cdot)$ can be similarly defined.

In the above definition, if the platform is computationally unbounded, the curator's privacy requirement simply says from platform's view, the protocol is (ϵ, δ)-DP where $\delta = \mathrm{negl}(\lambda)$ for all λ. Specifically, with $\delta = 0$, it becomes equivalent to information-theoretic ϵ-DP.

Any S2PC protocol for computing ϵ-DP mechanism satisfies ϵ-DP-S2PC. S2PC guarantees that a party learns nothing about the other party's input beyond the mechanism output while DP guarantees that the mechanism output satisfies ϵ-DP. Thus, platform's views are computationally ϵ-DP.

D.2 Online-Efficient Protocol with ROT

A DP-S2PC protocol can be constructed by combining the geometric garbling scheme in Figure 1 and OT. The curator generates a random vector $\mathbf{k} \in \mathbb{Z}_q^{p_i}$ for the platform's input wire i on `AuxMul` and the platform then obliviously obtains a random value $\mathbf{k}_j$ at the corresponding index j by running a 1-out-of-p_i OT.

However, in our scenario where the curator's input can be random but the platform's input needs to be chosen, we can utilize efficient sender random OT. This involves running $\text{ROT}_q^{p_i}$, where the curator obtains a random vector $\mathbf{k} \in \mathbb{Z}_q^{p_i}$ while the platform receives the random value $\mathbf{k}_j$ corresponding to their chosen index j.

With this approach, the curator uses the received random vector to garble the corresponding wire. This leads to a minor adjustment to the geometric garbling scheme in Figure 1. Specifically, we replace Line 4 of the garbling algorithm with the random vector $\mathbf{k}^i$ received as the the output of sender random OT. This modification does not affect the privacy of the geometric garbling scheme as the vector $\mathbf{k}^i$ received from sender random OT is uniformly chosen from $\mathbb{Z}_q^{p_i}$. The full construction is shown in Algorithm 1.

Our protocol achieves highly efficient online communication, where the curator only needs to transmit the encoding of her inputs during this phase. The communication cost for encoding each input $\mathbf{x}_i$ is $O(\log \mathbf{x}_i + \log \epsilon)$ with minimal overhead.

Algorithm 1 Online-efficient DP-S2PC protocol

Input: Data Curator (DC): $\mathbf{x}$. Platform Provider (PP): $w = \{w_i\}_{i\in[l]}$. **Common input:** privacy budget ϵ, ϵ_{out}, Geometric mechanism GM_f, geometric garbling scheme GG in Figure 1, ideal sender random OT functionality ROT.

Output: DC: $\perp$, PP: $\tilde{y} = \text{GM}_{\epsilon_{out}}(\mathbf{x}, w)$

Pre-processing phase:

1: DC and PP runs $(\mathbf{k}^i; Q_i) \leftarrow \text{ROT}_q^{p_i}(\perp; w_i)$ for each $i \in \texttt{AuxMul}$.

2: DC: runs $F, e, e_w, d \leftarrow GG.Gb(\epsilon, \text{GM}_{\epsilon_{out}})$ with Line 4 in Figure 1 replaced with $\mathbf{k}^i$; sends (F, d) to PP.

Online phase:

3: DC: runs $X \leftarrow GG.En(e, \mathbf{x})$ and sends X to PP.

4: PP: runs $Y \leftarrow GG.Ev(F, X, Q)$ and $\tilde{y} \leftarrow GG.De(d, Y)$.

Theorem D.1 (Privacy of Algorithm 1)**.** Under the semi-honest model, the protocol in Algorithm 1 satisfies ϵ-DP-S2PC with ϵ-private geometric garbling scheme and an ideal sender random OT functionality.

Proof. With the ideal sender random OT functionality, the curator learns nothing about the platform's private input w_i. Thus, the platform's privacy immediately holds. During the sender random OT protocol, the platform only learns the garbling of his input wire $L_i^{w_i}$ and nothing else. Thus, the curator's privacy follows the privacy of the geometric garbling scheme except for the modification to Line 4 in Figure 1. However, given the ideal sender random OT functionality, the vector $\mathbf{k}^i$ can be replaced with a random vector in $\mathbb{Z}_q^{p_i}$. Therefore, following the privacy of the geometric garbling scheme, the curator's privacy immediately holds. □

E The DP Query Answering as a Service

We present our practical application of DP-S2PC for the matrix mechanism that answers linear counting queries under DP, which we call *DP query answering as a service (DPQaaS)*.

Protocol Overview. Algorithm 2 shows DPQaaS, which employs our DP-S2PC protocol between the curator with a private data vector $\mathbf{x}$ and the platform with a private strategy matrix $\mathbf{S}$. The platform obtains DP measurements $\text{GM}_{\mathbf{S}}(\mathbf{x})$ at the end of the protocol. The platform learns nothing about $\mathbf{x}$ beyond what can be captured by ϵ-DP while the data curator learns nothing about the platform's strategy $\mathbf{S}$ in a cryptographic sense. Then, the platform performs our post-processing technique to reconstruct answers to all the input queries in $\mathbf{W}$, which provides an accuracy of $O(\epsilon)$.

E.1 Protocol Details

Algorithm 2 assumes both parties agree on privacy parameters $\epsilon_{in}, \epsilon_g, \epsilon_{out}$, where $\epsilon_{in} + \epsilon_g + \epsilon_{out} = \epsilon$. We denote q as the largest integer supported during the protocol. The protocol begins with the platform

Algorithm 2 DPQaaS

Input: Data curator (DC): data vector $\mathbf{x} \in \mathbb{N}^n$. Platform provider (PP): query matrix $\mathbf{W} \in \mathbb{R}^{l \times n}$. **Common input:** privacy budget $\epsilon = \epsilon_{in} + \epsilon_g + \epsilon_{out}$, scaling factor t, Geometric mechanism GM, ideal sender random OT functionality ROT.

Output: DC: $\perp$. PP: ϵ-DP query answers $\tilde{\mathbf{a}} \in \mathbb{R}^l$.

Pre-processing phase:

1: PP: $\mathbf{S} \leftarrow \mathrm{OPT}_t(\mathbf{W}) \in \mathbb{Z}_{t+1}^{m \times n}$ (instantiate with [38]);

2: DC $\leftrightarrow$ SA: $(\{\mathbf{k}^{i,j}\}_{i\in[m],j\in[n]}; \mathbf{Q}) \leftarrow mn \times \mathrm{ROT}_q^{t+1}(\perp; \mathbf{S})$;

3: DC: $\mathbf{r} \leftarrow \mathrm{Geo}(\frac{1}{\epsilon_{in}})^n, \mathbf{Z} \leftarrow \mathrm{Geo}(\frac{t}{\epsilon_g})^{m\times n}, \mathbf{b} \leftarrow \mathrm{Geo}(\frac{t}{\epsilon_{out}})^m$;

4: $\quad \mathbf{G}^s = \mathbf{1}_m(s \cdot \mathbf{r})^T + (\mathbf{k}_{s+1}^{i,j}) - \mathbf{Z}, \forall s \in \mathbb{Z}_{t+1}$;

5: $\quad \mathbf{d} = \mathbf{Z}\mathbf{1}_n - \mathbf{b}$;

6: DC $\rightarrow$ PP: $\mathbf{G}^0, \ldots, \mathbf{G}^t$ and $\mathbf{d}$;

Online phase:

7: DC: $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{r}$;

8: DC $\rightarrow$ PP: $\tilde{\mathbf{x}}$;

9: PP: $\tilde{\mathbf{C}} = \mathbf{S} \circ (\mathbf{1}_m \tilde{\mathbf{x}}^T) + \mathbf{Q} - (\mathbf{G}_{i,j}^{\mathbf{S}_{i,j}})$;

10: $\quad \tilde{\mathbf{o}} = \tilde{\mathbf{C}}\mathbf{1}_n$;

11: $\quad \tilde{\mathbf{y}} = \tilde{\mathbf{o}} - \mathbf{d}$;

Post-processing phase:

12: PP: $\tilde{\mathbf{a}}^{\mathbf{I}} = \mathbf{W}\tilde{\mathbf{x}}$, $\tilde{\mathbf{a}}^{\mathbf{B}} = \mathbf{W}\mathbf{B}^+ vec(\tilde{\mathbf{C}})$, $\tilde{\mathbf{a}}^{\mathbf{S}} = \mathbf{W}\mathbf{S}^+\tilde{\mathbf{y}}$; ▷ $\mathbf{B} = diag(\mathbf{S})$

13: $\quad \tilde{\mathbf{a}}_i = \frac{\sum_{\mathbf{A}\in\{\mathbf{I},\mathbf{B},\mathbf{S}\}} \tilde{\mathbf{a}}_i^{\mathbf{A}}/\mathrm{Var}(\tilde{\mathbf{a}}_i^{\mathbf{A}})}{\sum_{\mathbf{A}\in\{\mathbf{I},\mathbf{B},\mathbf{S}\}} 1/\mathrm{Var}(\tilde{\mathbf{a}}_i^{\mathbf{A}})}, \forall i \in [l]$.

selecting the best strategy matrix $\mathbf{S}$ through an optimization procedure, OPT, instantiated with state-of-the-art techniques [38] (Line 1). The strategy matrix $\mathbf{S}$ is quantized to $\mathbb{Z}_{t+1}^{m\times n}$ using a scaling factor t, detailed below.

We then adapt the DP-S2PC protocol from Algorithm 1 to compute the Geometric mechanism $\mathrm{GM}_{\epsilon_{out}}(\mathbf{x}, \mathbf{S}) = \mathbf{S}\mathbf{x} + \mathrm{Geo}(||\mathbf{S}||_1/\epsilon_{out})^m$. The matrix-vector product is represented as an arithmetic circuit to be garbled. Although we use linear algebraic expressions for garbling, the underlying operations remain the same as gate expressions. Both the data curator and the platform invoke the $mn \times \mathrm{ROT}_q^{t+1}$-oracle (Line 2) (Lines 3-11). At the end of this protocol, the platform obtains the ϵ_{out}-DP output $\tilde{\mathbf{y}} = \mathrm{GM}_{\epsilon_{out}}(\mathbf{x}, \mathbf{S})$. This protocol satisfies ϵ-DP-S2PC with the privacy budget allocation of $\epsilon_{in} + \epsilon_g + \epsilon_{out} = \epsilon$.

Given the DP mechanism outputs $\tilde{\mathbf{y}}$, we can reconstruct answers to all queries in $\mathbf{W}$ by computing $\mathbf{W}\mathbf{S}^+\tilde{\mathbf{y}}$, achieving an accuracy of $O(\epsilon_{out})$. However, we will show that this accuracy can be further improved to $O(\epsilon)$, where $\epsilon_{out} < \epsilon$. This reconstruction step does not degrade privacy due to the post-processing theorem.

Quantization. The strategy query matrix $\mathbf{S}$ consists of normalized, non-negative values satisfying $||\mathbf{S}||_1 = 1$. Since our DP-2PC protocol works over integers, we apply a scaling factor t to the strategy matrix. This ensures that the resulting strategy matrix is represented in integers within the range $[0, t]$ and $||\mathbf{S}||_1 = t$. This scaling factor impacts the precision of the original strategy matrix. In our experiments, a value of $t = 100$ was sufficient to represent the original values without significantly degrading accuracy.

E.2 Boosting Accuracy

We describe our reconstruction technique in Line 12-13 in Algorithm 2. Our approach provides DP query answers with an accuracy of $O(\epsilon)$ by combining multiple independent measurements produced from garbled circuits as artifacts.

Our key observation is that during the 2PC protocol, the platform obtains three distinct types of DP query measurements. First, the encoding of inputs $\tilde{\mathbf{x}}$ can be viewed as the ϵ_{in}-DP measurements to the identity query matrix $\mathbf{I}$, i.e., $\mathbf{I}\mathbf{x} + \mathrm{Geo}(1/\epsilon_{in})^n$. Second, the DP wire labels obtained by evaluating multiplication gates can be viewed as ϵ_g-DP measurements of an mn by n query matrix $\mathbf{B}$, i.e., $\mathbf{B}\mathbf{x} + \mathrm{Geo}(||\mathbf{B}||_1/\epsilon_g)^{mn}$. Here, $\mathbf{B}$ is derived by vertically stacking a diagonal matrix of each row vector $\mathbf{S}_i$, denoted as $diag(\mathbf{S}_i)$. We

abuse the notation and write $\mathbf{B} = diag(\mathbf{S})$. Note that the sensitivity of the two query matrices are the same, i.e., $||\mathbf{B}||_1 = ||\mathbf{S}||_1 = t$. Lastly, the third type of measurement is the protocol output itself, i.e., $\mathbf{Sx} + \text{Geo}(||\mathbf{S}||_1/\epsilon_{out})^m$.

Using these three query matrices that support the input query matrix $\mathbf{W}$, we reconstruct independent answers for all input queries in Line 12: 1) ϵ_{in}-DP answers $\tilde{\mathbf{a}}^{\mathbf{I}}$ for the identity matrix $\mathbf{I}$, 2) ϵ_g-DP answers $\tilde{\mathbf{a}}^{\mathbf{B}}$ for the stacked diagonal matrix $\mathbf{B}$, and 3) ϵ_{out}-DP answers $\tilde{\mathbf{a}}^{\mathbf{S}}$ for the strategy matrix $\mathbf{S}$.

We then combine these reconstructed answers using the inverse-variance weighting method [30]. Specifically, let y be a sequence of independent answers to each query vector $\mathbf{W}_i$, denoted as, $y = \tilde{\mathbf{a}}_i^{\mathbf{I}}, \tilde{\mathbf{a}}_i^{\mathbf{B}}, \tilde{\mathbf{a}}_i^{\mathbf{S}}$. The inverse-variance weighted average is then computed as: $\bar{y} = \frac{\sum_j y_j w_j}{\sum_j w_j}$, with weights $w_j = \frac{1}{\text{Var}(y_j)}$. This combined estimate has the least variance among all weighted averages, given as $\text{Var}(\bar{y}) = \frac{1}{\sum_j 1/\text{Var}(y_j)}$. Here, Var denotes the variance of a random variable. The variance of an estimated answer $\mathbf{a}_i^{\mathbf{A}}$ for a matrix $\mathbf{A}$ corresponds to $\text{Err}(\mathbf{W}_i, \mathbf{A})$ (in Definition A.4). Thus, the error of the final estimated answers using our reconstruction technique is derived as follows.

Theorem E.1 (Query Error of DPQaaS)**.** Given a query matrix $\mathbf{W}$ and strategy query matrix $\mathbf{S}$, the error of the answer to each query $\mathbf{W}_i$ in Algorithm 2 is:

$$\text{DPQaaS-Err}(\mathbf{W}_i, \mathbf{S}) = \frac{1}{\frac{1}{\text{Err}_{\epsilon_{in}}(\mathbf{W}_i, \mathbf{I})} + \frac{1}{\text{Err}_{\epsilon_g}(\mathbf{W}_i, \mathbf{B})} + \frac{1}{\text{Err}_{\epsilon_{out}}(\mathbf{W}_i, \mathbf{S})}},$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{B} = diag(\mathbf{S})$.

E.3 Privacy, Accuracy and Efficiency Tradeoffs

We analyze the impact of privacy loss parameters $\epsilon_{in}, \epsilon_g, \epsilon_{out}$ on accuracy of query answers and efficiency of our protocol in Algorithm 2. We show tradeoffs between:1) privacy loss ϵ_{in} and online communication cost, 2) privacy loss $\epsilon_g + \epsilon_{out}$ and decoding information size, and 3) privacy loss $\epsilon_{in} + \epsilon_g + \epsilon_{out}$ and query accuracy.

Online Communication Cost. The online communication cost of our protocol is affected by the size of input encoding, which is proportional to the bit-length of the random noise added to the vectorized counts. Encoding of $\mathbf{x}$ results in: $\sum_{i=1}^{n} \log |\mathbf{x}_i + \mathbf{r}_i|$ bits, where $\mathbf{r}_i \leftarrow \text{Geo}(1/\epsilon_{in})^n$. Larger absolute noise values can increase encoding size, leading to a tradeoff between privacy and communication cost. Smaller values of ϵ_{in} may result in larger noise, i.e., longer bit-lengths, thus resulting in a larger encoding size. However, larger values of ϵ_{in} yield smaller noise, effectively reducing the overhead of online communication cost.

Decoding Information Size We examine the relationship between the privacy loss parameter $\epsilon_g + \epsilon_{out}$ and the size of the decoding information. Recall that the decoding information d for each output is computed as $d = (\sum_{j=1}^{n} z_j) - b$, where $z_j \leftarrow \text{Geo}(t/\epsilon_g)$ and $b \leftarrow \text{Geo}(t/\epsilon_{out})$. The variance of d increases as the domain size n increases and the values of ϵ_g and ϵ_{out} decrease. For example, with a domain size of 100, $\epsilon_g = 1.0, \epsilon_g = 1.0$ result in an expected bit-length of 10 bits for the decoding value d. Even with small privacy parameters, such as $\epsilon_g = 0.001, \epsilon_g = 0.001$, the cost is only 20 bits, significantly less than BMR garbling, which requires approximately 50 kilobits for 64-bit arithmetic circuits.

The size of decoding information affects the offline communication cost. When online efficiency is of greater importance, the choice of privacy loss parameters for decoding information size may be less critical.

Accuracy. In our reconstruction step, we have improved the accuracy of the ϵ_{out}-DP mechanism output by combining three independent DP measurements, resulting in $(\epsilon_{in} + \epsilon_g + \epsilon_{out})$-DP query answers. While increasing ϵ_{in} and ϵ_g for a fixed ϵ_{out} can enhance accuracy, there is a tradeoff with privacy. Importantly, the ϵ_{out}-DP measurements with respect to the strategy query $\mathbf{S}$ remain the primary contributors to the final accuracy, as they are obtained under the optimal strategy. Therefore, to achieve high accuracy, one should prioritize considering the value of ϵ_{out} first, and then consider ϵ_{in} and ϵ_g as secondary values. We will verify this in our experiments.

Setting Privacy Parameters. Overall, one way to set $\epsilon_{in}, \epsilon_g, \epsilon_{out}$ would be: first set ϵ_{out} as in traditional DP mechanisms, taking into account the tradeoff between privacy and accuracy; then set ϵ_{in} and ϵ_g according to how much privacy loss one is willing to accept in exchange for desired levels of online communication performance and decoding key size, respectively. In terms of online efficiency, prioritizing ϵ_{in} over ϵ_{out} is recommended.

E.4 Correctness Proof

We show the correctness of the DP-S2PC protocol in Algorithm 2.

Proof. We show that the platform correctly outputs $\tilde{\mathbf{y}} = \mathrm{GM}_{\epsilon_{out}}(\mathbf{x}, \mathbf{S})$ at the end of the protocol in Line 11:

$$
\begin{aligned}
\tilde{\mathbf{y}} &= \tilde{\mathbf{c}} - \mathbf{d} \\
&= \tilde{\mathbf{O}}\mathbf{1}_n - \mathbf{Z}\mathbf{1}_n + \mathbf{b} \\
&= \left[\mathbf{S} \circ (\mathbf{1}_m \tilde{\mathbf{x}}^T) + \mathbf{Q} - \left(\mathbf{G}_{i,j}^{\mathbf{S}_{i,j}}\right)\right] \mathbf{1}_n - \mathbf{Z}\mathbf{1}_n + \mathbf{b} \\
&= \left[\mathbf{S} \circ \left(\mathbf{1}_m \left(\mathbf{x} + \mathbf{r}\right)^T\right) + \left(\mathbf{k}_{\mathbf{S}_{i,j}+1}^{i,j}\right) - \left(\mathbf{S} \circ (\mathbf{1}_m \mathbf{r}^T) + \left(\mathbf{k}_{\mathbf{S}_{i,j}+1}^{i,j}\right) - \mathbf{Z}\right)\right] \mathbf{1}_n \\
&\quad - \mathbf{Z}\mathbf{1}_n + \mathbf{b} \\
&= \mathbf{S} \circ (\mathbf{1}_m \mathbf{x}^T)\mathbf{1}_n + \mathbf{b} \\
&= \mathbf{S}\mathbf{x} + \mathrm{Geo}(t/\epsilon_{out})^m \\
&= \mathbf{S}\mathbf{x} + \mathrm{Geo}(||\mathbf{S}||_1/\epsilon_{out})^m.
\end{aligned}
$$

This completes the proof by substitution. □

E.5 Privacy Proof

We show Algorithm 2 satisfies the definition of ϵ-DP-S2PC (in Definition D.1). The privacy proof is similar to that of Theorem D.1 where the privacy of Algorithm 2 follows the privacy of the geometric garbling scheme and the privacy of sender random OT. A key distinction is that we provide the geometric garbling scheme in Figure 1 with a tighter privacy budget allocation of $\epsilon = \epsilon_{in} + \epsilon_g + \epsilon_{out}$. In this section, we show that even with the tight budget allocation, the geometric garbling scheme for computing $\mathrm{GM}(\mathbf{x}, \mathbf{S})$ in Algorithm 2 satisfies the ϵ-privacy definition. Note that the reconstruction step in Algorithm 2 does not degrade privacy due to the post-processing theorem.

Theorem E.2 (Privacy of Algorithm 2)**.** Under the semi-honest model, DPQaaS with an ideal sender random OT satisfies ϵ-DP-S2PC.

Proof. The proof follows that in Appendix C.6. All we need to show is that a mechanism $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x})$ for Algorithm 2 satisfies ϵ-DP.

Recall that $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x})$ is defined as a mechanism that outputs garbled input, garbled output of every evaluator-half-gate and the output itself. These correspond to $\tilde{\mathbf{x}}$, $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{y}}$ in Algorithm 2. Thus, the mechanism $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x})$ can be derived as a composition of the following three mechanisms, denoted as $\mathrm{VIEW}_{\mathcal{A}}(\mathbf{x}) = (\mathcal{E}, \mathcal{G}, \mathcal{D})$.

- $\mathcal{E}(\mathbf{x})$: compute $\tilde{\mathbf{x}} = \mathbf{I}_n\mathbf{x} + \mathrm{Geo}(1/\epsilon_{in})^n$; return $\tilde{\mathbf{x}}$.
- $\mathcal{G}(\mathbf{x})$: compute $\tilde{\mathbf{c}} = \mathbf{B}\mathbf{x} + \mathrm{Geo}(||\mathbf{S}||_1/\epsilon_g)^{mn}$ and reshape it into m by n matrix $\tilde{\mathbf{C}}$; return $\tilde{\mathbf{C}}$.
- $\mathcal{D}(\mathbf{x})$:compute $\tilde{\mathbf{y}} = \mathbf{S}\mathbf{x} + \mathrm{Geo}(||\mathbf{S}||_1/\epsilon_{out})^m$; return $\tilde{\mathbf{y}}$.

In the above mechanisms, $\mathbf{I}_n$ denotes the identity matrix of size n and $\mathbf{1}_n$ denotes the all-ones vector of length n. In addition, the matrix $\mathbf{B}$ is derived by constructing a diagonal matrix of each row vector $\mathbf{S}_i$ and

stacking the resulting diagonal matrices vertically:

$$\mathbf{B} = \begin{pmatrix} diag(\mathbf{S}_1) \\ diag(\mathbf{S}_2) \\ \vdots \\ diag(\mathbf{S}_m) \end{pmatrix}$$

where $diag(\cdot)$ denotes a vector-to-matrix diagonal operator. We may call the matrix $\mathbf{B}$ as a *stacked diagonal matrix*.

One can view the above mechanisms $\mathcal{E}, \mathcal{G}, \mathcal{D}$ as mechanisms for answering the identity query matrix $\mathbf{I}$, the stacked diagonal query matrix $\mathbf{B}$ and the input query matrix $\mathbf{S}$. Thus, from the privacy of the Geometric mechanism, the mechanisms $\mathcal{E}$, $\mathcal{G}$ and $\mathcal{D}$ satisfy ϵ_{in}-DP, ϵ_g-DP, and ϵ_{out}-DP, respectively. Note that for the mechanism $\mathcal{G}$, the sensitivity of the matrix $\mathbf{B}$ is the same as that of $\mathbf{S}$, i.e., $||\mathbf{B}||_1 = ||\mathbf{S}||_1 = t$ ($\mathbf{B}$ is derived by reformatting $\mathbf{S}$ without affecting its sensitivity). In addition, from the post-processing theorem, the garbled output vector $\tilde{\mathbf{o}}$ is derived without degrading privacy.

Hence, from the sequential composition theorem, the mechanism $\text{VIEW}_{\mathcal{A}}$ satisfies ϵ-DP, where $\epsilon = \epsilon_{in} + \epsilon_g + \epsilon_{out}$. Thus, the geometric garbling scheme in Algorithm 2 satisfies ϵ-privacy.

□

E.6 Handing High-Dimensional Queries

We extend Algorithm 2 to effectively compute high-dimensional queries. Consider d-dimensional datasets, a query matrix $\mathbf{S}$ can be implicitly represented as Kronecker products of d query matrices [38], denoted as $\mathbf{S} = \mathbf{S}^1 \otimes \cdots \otimes \mathbf{S}^d$. Here, $\mathbf{S}^i$ is a m_i by n_i query matrix over the i-th domain. Given a d-dimensional data vector $\mathbf{x}$ of length $\prod_{i=1}^{d} n_i$, query answers to the matrix $\mathbf{S}$ can be expressed as: $\mathbf{y} = (\mathbf{S}^1 \otimes \cdots \otimes \mathbf{S}^d)\mathbf{x}$. We can effectively compute this without performing the Kronecker products by utilizing the following property: $(\mathbf{A} \otimes \mathbf{B})vec(\mathbf{X}) = vec(\mathbf{B}\mathbf{X}\mathbf{A}^{\mathbf{T}})$. The expression on the right allows us to avoid materializing the potentially large matrix $\mathbf{A} \otimes \mathbf{B}$. Thus, by setting $\mathbf{A} \leftarrow \mathbf{S}^1 \otimes \cdots \otimes \mathbf{S}^{d-1}, \mathbf{B} \leftarrow \mathbf{S}^d$, we can compute matrix-vector products for matrices of the forms $\mathbf{S}^1 \otimes \cdots \otimes \mathbf{S}^d$ efficiently as shown in Algorithm 3. With this procedure, the time and space complexity of computing the query answers is $O(n^d)$ and $O(dn^{d+1})$, where we assume $\mathbf{S}^i \in \mathbb{R}^{n \times n}$ for all i for simplicity. A naive approach requires $O(n^{2d})$ time and space.

Algorithm 3 Vector-Kronecker product multiplication

Input: Query matrix $\mathbf{S}$ of an implicit representation $\mathbf{S}^1 \otimes \cdots \otimes \mathbf{S}^d$, where $\mathbf{S}^i \in \mathbb{R}^{m_i \times n_i}$, data vector $\mathbf{x}$
Output: Query answers $\mathbf{y} = \mathbf{S}\mathbf{x}$

1: $\mathbf{X}^d \leftarrow \mathbf{x}$
2: $n_0 \leftarrow 1, m_{d+1} \leftarrow 1$.
3: **for** $i = d \ldots 1$ **do**
4: Reshape $\mathbf{X}^i$ into a size n_i by l_i, where $l_i = \Pi_{j=0}^{i-1} n_j \times \Pi_{j=i+1}^{d+1} m_j$
5: $\mathbf{X}^{i-1} = \mathbf{S}^i \mathbf{X}^i$
6: **end for**
7: $\mathbf{y} \leftarrow vec(\mathbf{X}^0)$.

The procedure in Algorithm 3 requires a chain of d matrix multiplications and our geometric scheme can support such computations. Each iteration requires a matrix multiplication, which can be done similarly as Algorithm 2. The only difference is that there are multiple iterations of matrix multiplication, meaning that a privacy budget is divided among the iterations following the sequential composition theorem. In addition, we make a key observation that each iteration i measures the following query answers that are revealed to the adversarial evaluator:

$$(\mathbf{I}_{n_1} \otimes \cdots \otimes \mathbf{I}_{n_{i-1}} \otimes \mathbf{S}^i \otimes \cdots \otimes \mathbf{S}^d)\mathbf{x}.$$

Thus, each fan-in-two multiplication gate is assigned with DP-noise from $\text{Geo}(\Delta_i d/\epsilon_g)$ with the sensitivity $\Delta_i = ||\mathbf{I}_{n_1} \otimes \cdots \otimes \mathbf{I}_{n_{i-1}} \otimes \mathbf{S}^i \otimes \cdots \otimes \mathbf{S}^d||_1$.

F Performance Evaluation

F.1 Experimental Setup

Metrics

We measure both the offline-online communication and computational cost of the 2PC protocol. Communication costs are measured in terms of network traffic, indicating the volume of data exchanged between two parties. Any computation that can be performed independently of datasets is considered offline preprocessing.

To measure the accuracy of answers to query matrix $\mathbf{W}$, we compute a normalized version of expected squared error in Theorem E.1 across all queries in $\mathbf{W}$, denoted as $\text{RMSE}_{\epsilon_{in},\epsilon_g,\epsilon_{out}}(\mathbf{W},\mathbf{S})$. Similarly, we denote $\text{RMSE}_{\epsilon}(\mathbf{W},\cdot)$ as the normalized version of error metric in Definition A.4.

Data and Queries

We use five datasets from the DPBench benchmark [27], regularly employed in DP literature: ADULT-FRANK, HEPTH, INCOME, NETTRACE, and PATENT. The maximum domain size for these datasets is 4096. We generate smaller versions of each dataset by grouping adjacent histogram buckets, resulting in domain sizes of $\{128, 256, 524, 1028\}$.

We consider two workloads for input queries $\mathbf{W}$: prefix workload and all-range workload. A prefix workload consists of predicates that define the CDF and has the form of a lower triangular n by n matrix, with each row representing a prefix sum. An all-range workload encodes range queries of the form $[i, j]$ for $1 \leq i \leq j \leq n$. For the given input workload, we derive a strategy matrix $\mathbf{S}$ using the state-of-the-art method called a p-Identity strategy [38]. The matrix is normalized to the range of $[0, 1]$ and then represented as fixed-point numbers with a scaling factor. This factor determines the tradeoff between precision and protocol performance. In our experiment, we use a scaling factor of 100, which only degrades query accuracy by less than 1%.

Competing techniques

We evaluate our techniques against several baselines under a fixed privacy budget of ϵ.
Insecure Baseline. The first benchmark is an insecure method, where the data curator sends the raw data directly to the platform, who then performs the Geometric mechanism under ϵ with a p-Identity strategy matrix $\mathbf{S}$. While this approach does not protect data privacy against the platform, it offers optimal performance with an optimal error of $\text{RMSE}_{\epsilon}(\mathbf{W},\mathbf{S})$. This comparison allows us to assess the efficiency overhead incurred by DPQaaS compared to an insecure, yet optimal, method. For accuracy, we also consider a variant of the insecure baseline with an identity strategy matrix $\mathbf{I}$, resulting in an error of $\text{RMSE}_{\epsilon}(\mathbf{W},\mathbf{I})$. The accuracy of DPQaaS is expected to be no worse than this.
BMR Garbling. Our second benchmark is the state-of-the-art arithmetic garbled circuit technique, BMR garbling (BMR) [8], which incorporates numerous optimization techniques. With a BMR-based protocol, the platform obtains ϵ-DP measurements for the quantized strategy matrix. The resulting query answers have the error of $\text{RMSE}_{\epsilon}(\mathbf{W}, \mathit{quantize}(\mathbf{S}))$.
One-time Pad Encoding. We also evaluate a variant of geometric garbling that employs one-time pad (OTP) instead of DP encoding. Instead of adding DP noise corresponding to ϵ_{in} and ϵ_g, we mask the relevant values by adding uniform random numbers. In practice, without violating DP, we cannot inspect the actual data to determine the OTP size, so we use 32-bits and 64-bits, assuming these are sufficient to support the maximum count in $\mathbf{x}$. This comparison helps us evaluate the effectiveness of DP wire labels against OTP in terms of online communication costs. This approach provides the same query accuracy as BMR garbling.

Implementation details. We conduct all experiments on a single MacBook Air (M2 chip with 16GB RAM). All techniques including the benchmarks are implemented in Rust. We use a 64-bit integer representation and, therefore, set $q = 2^{64}$ in our DP-S2PC protocol. We consider a scaling factor of $t = 100$. To implement sender random OT in Algorithm 2, we use the sender random 1-out-of-2 OT extension of [4] with the base-OT of [17].

We use the fancy-garbling library [6, 23] to implement BMR garbling. This library incorporates numerous practical optimization techniques and supports many variants of OT. To establish OT, we use the same state-of-the-art OT extension protocol [4] as well as the same base-OT of [17]. To prevent overflow during computation, we set a hyperparameter $k = 16$, which specifies the number of primes used in CRT representation. The library provides a communication channel that enables data transmission at the byte level, which we employ to implement our benchmarks.

F.2 Empirical Results

Computational and Communication Cost

We analyze the computational and communication costs of our DP-S2PC protocol, benchmarked against the competing techniques. We employ a prefix workload with various domain sizes, resulting in p-Identity strategy matrices of dimensions: $\{136\times128, 266\times256, 527\times512, 1054\times1024\}$. We set a fixed privacy budget of 1.0 with a budget allocation of $\epsilon_{in} = 0.09, \epsilon_g = 0.01, \epsilon_{out} = 0.9$. Our evaluation uses the ADULTFRANK dataset and average the cost across five runs. Varying datasets or queries do not affect the performance of BMR garbling.

Figure 3 shows that our technique significantly outperforms BMR garbling in terms of runtime and network traffic. For instance, with a domain size of 1024, BMR requires 10 minutes and 488 MB of network traffic to execute the 2PC protocol. In contrast, our protocol completes in 3 minutes and only consumes 32 MB of network traffic —an improvement of approximately 3 and 15 times for runtime and network traffic.

In addition, Figure 4 shows a comparison of the protocol performance against the other methods, including the insecure baseline and the 32-bit OTP variant. Despite offering significant performance improvements against BMR, there remains an overhead compared to the insecure baseline - an essential privacy tradeoff. However, in online performance, DPQaaS requires only a minimal network traffic (e.g., 2KB for a domain size of 1024), matching the insecure baseline, thus achieving optimal efficiency. We note that the BMR implementation does not focus on minimizing online performance. Although the total runtime and network traffic show negligible difference between our DPQaaS and the OTP-based DPQaaS, the consistent efficiency of DPQaaS in online network traffic is demonstrable, further discussed below.

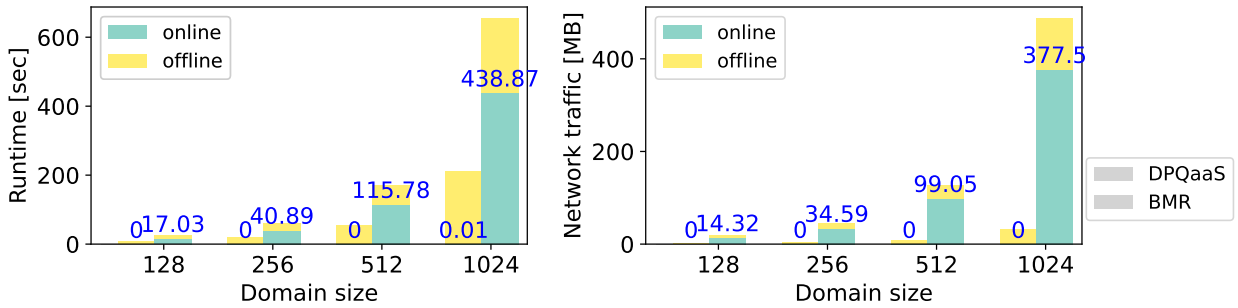
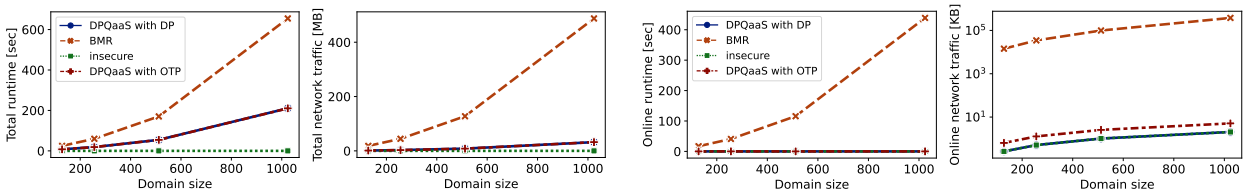


Figure 3: Offline-online network traffic cost (right) and runtime (left) of our protocol, compared to BMR garbling. Values in blue are online costs. Costs less than 10KB or 10ms are reported as 0.

Very Low Online Communication.

In this section, we investigate the efficiency of the online communication cost of our protocol and how the privacy loss parameter ϵ_{in} affects the cost. Figure 5 shows the average total online communication cost over 100 trials for various values of ϵ_{in}. We used datasets with a domain size of 1024. We did not include results for other domain sizes, as their performance showed similar results. To benchmark our technique, we compare it against the two baselines: (i) the insecure baseline and (ii) our technique with 32-bit and 64-bit OTP encodings. We do not compare against BMR garbling due to its high encoding overhead.

Our results indicate that decreasing ϵ_{in} increases the encoding size, leading to a larger overhead. Conversely, increasing ϵ_{in} results in an optimal overhead as the noise added to encode raw counts gets smaller,



(a) Total performance cost

(b) Online performance cost

Figure 4: Performance comparison among different protocols when varying domain sizes. The online network traffic is reported on a logarithmic scale. The total performance cost combines both online and offline costs.

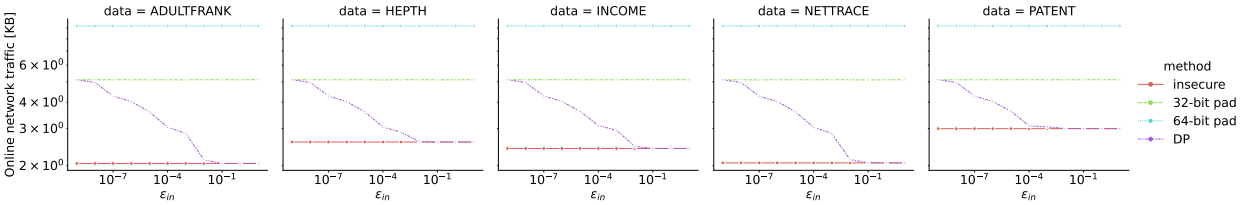


Figure 5: Online network traffic comparison among different protocols across various datasets when varying the privacy loss parameter ϵ_{in}.

resulting in lower communication costs. Moreover, our DP encoding provides a tighter bound on the overhead and can outperform the fixed-size OTP, especially when determining the upper bound of the input data in practice without violating DP is challenging. For instance, with $\epsilon_{in} = 1.0$, the DP encoding cost has almost no overhead and is 2 and 4 times smaller than the 32-bit and 64-bit OTP encoding. Even with small $\epsilon_{in} = 0.001$, the DP encoding cost is only 1.0 to 1.3 times larger than the insecure baseline, while still being more cost-effective than OTP encoding.

Importantly, the overhead can vary depending on the dataset's distribution. Our DP encoding achieves greater benefits on datasets like ADULTFRANK and NETTRACE, where over 96% of the data consist of zero counts. We also observe that our protocol becomes comparable to the 32-bit OTP benchmark when $\epsilon_{in} = 1e\text{-}8$, an extremely small value. However, if there is public background knowledge about the small upper bounds of counts, then DP encoding may not offer significant advantages over fixed-size OTP.

Accuracy

We conducted experiments to evaluate the accuracy of our approach, focusing on prefix and all-range queries on a domain size of 128. We used the root mean square error (RMSE) as the error metric, which is independent of the data. We examined different values of ϵ_{in}, ϵ_g, and ϵ_{out} to achieve comparable accuracy against an optimal but insecure baseline under a fixed privacy loss of 1.

The results, summarized in Table 2, show that our approach can achieve near-optimal accuracy by allocating larger values to ϵ_{out}. For example, an allocation of $\epsilon_{in} = 0.009, \epsilon_g = 0.001, \epsilon_{out} = 0.99$ results in an accuracy degrade of less than 2%. A more aggressive allocation with $\epsilon_{in} = 9e\text{-}5, \epsilon_g = 1e\text{-}5, \epsilon_{out} = 0.9999$ achieves the same accuracy as the BMR garbling solution while requiring an order of magnitude less communication cost. Note that all secure solutions have a small accuracy reduction compared to an insecure but optimal approach due to quantization.

Our recommended privacy loss configuration prioritizes accuracy first and online communication cost second, with a larger budget allocated to ϵ_{in} than ϵ_g. The query error is determined by what measurement queries are used to answer (see Definition A.4). We observed that $\text{RMSE}_\epsilon(\mathbf{W}, \mathbf{S}) < \text{RMSE}_\epsilon(\mathbf{W}, \mathbf{I}) < \text{RMSE}_\epsilon(\mathbf{W}, \mathbf{B})$. Thus, for a fixed privacy budget, assigning most to ϵ_{out} for the strategy matrix $\mathbf{S}$ and then dividing the remaining between ϵ_{in} and ϵ_g accordingly would result in high accuracy. However, if the remaining budget for $\epsilon_{in}, \epsilon_{out}$ is small, different allocations between the two parameters had a minor effect on final accuracy.

Table 2: Comparing errors under a fixed privacy loss budget of 1.0 with various allocations of the privacy budget $(\epsilon_{in}, \epsilon_g, \epsilon_{out})$. An asterisk (*) indicates quantization was applied.

Configuration			Query error	
Method	Strategy	Privacy loss	Prefix	All-Range
insecure	Identity	1.0	10.8	8.93
insecure	p-Identity	1.0	6.08	6.49
BMR/OTP	p-Identity*	1.0	6.13	6.51
		(0.009, 0.001, 0.99)	6.20	6.58
Ours	p-Identity*	(0.09, 0.01, 0.9)	6.85	7.27
		(0.1, 0.05, 0.85)	7.27	7.71

G Related Work

Considerable theoretical and applied research has been focused to optimizing garbled circuits. The current state-of-the-art technique for arithmetic garbled circuits is BMR garbling [8], which builds upon Free-XOR [33] and Half-gates [48] to efficiently handle addition and multiplication operations. These are key building blocks to perform matrix multiplication. However, even with highly optimized garbled circuits, there is an inherent crypotgraphic overhead. For instance, in a 64-bit arithmetic circuit, BMR garbling encodes each wire into a 2048-bit wire label. A recent garbling technique relies on Paillier encryption, requiring a 4096-bit label [7].

Many real-world applications find that relying solely on cryptographic primitives is inefficient, prompting numerous prior works to focus on designing secure and efficient protocols for specific functions and applications [14, 31, 41, 36, 43]. Our garbling technique is also tailored to our application and do not work for arbitrary functions like general garbled circuits.

Several works have investigated using DP to enhance cryptographic protocol efficiency while maintaining meaningful security, e.g., [32, 13, 28, 37, 9, 44]. A comprehensive survey of the intersection of cryptography and DP is provided by Wagh et al. [45]. Prior work has integrated DP to reduce the computational cost of cryptographic protocols, especially in query processing [9] and record linkage problems [28]. Their approach involves using DP cardinality to hide the size of intermediate results, avoiding the use of an exhaustive padding strategy that results in significant computational overhead. However, these solutions still depend on cryptographic primitives such as secret sharing, garbled circuits, and/or homomorphic encryption.

While our approach to DP-S2PC allows one party to learn DP outputs, prior work on private set intersection considers scenarios where *both* parties learn DP functions of each other's inputs [26]. The simultaneous consideration of what to compute and how to compute it was examined by Beimel et al. [11], who also offered a DP-style definition for multi-party computation (MPC). However, they found that substituting traditional MPC with DP may not always yield desired benefits. Subsequent research has explored the concept of "differentially oblivious" algorithms, initiated with DP oblivious RAM [16, 12, 22, 34]. These studies have demonstrated asymptotic efficiency improvements in certain cases.