An Efficient Gaussian Mechanism under Continual Observation

Rasmus Pagh[1] and Sia Sejer[1]

[1]BARC, University of Copenhagen

February 19, 2026

Abstract

Consider a data set $X = \{(u_i, x^{(i)}) : i \in [n]\}$, where $u_i \in \mathbb{Z}$, $x^{(i)} \in \mathbb{R}^k$ with $\left\|x^{(i)}\right\|_2 \leq \Delta_2$ for all i. Two data sets are neighbouring if they have a symmetric difference of at most 1. We consider the private release of $A^{(t)} = \sum_{i\,:\,u_i \leq t} x_i$ for $t = 1, \ldots, T$, i.e., the continual release of a vector A that is incremented by a vector x_i at time u_i.

Standard continual counting techniques allow us to do this with additive noise vectors with ℓ_∞-norm polylog(T), computed in time $\mathcal{O}(kT)$. Motivated by applications in private sketching, we consider the setting where only a *subset* of entries in $A^{(t)}$ is released at time step t and show that we can sample an entry in a given noise vector in *constant time*, simulating the distribution of standard tree-based continual counting using Gaussian noise.

1 Introduction

Differential privacy provides a strong information-theoretic guarantee of privacy with a tunable trade-off between privacy and *utility*, measured by the magnitude of noise added to query answers. Thus, it is relevant in settings where data points may reveal sensitive information about individuals. In this paper, we will present a reduction for maintaining a k-dimensional vector $A \in \mathbb{R}^k$ under continual observation. It is easy to make a single instance of a vector private by adding, say, Gaussian noise to the result. In the *continual observation* setting, queries are interleaved with updates, and we want the whole history of query answers to be differentially private. A naive approach would be to instantiate k versions of the classical binary mechanism using Gaussian noise Dwork, Naor, Pitassi, and Rothblum [2010]. This approach, which updates all k noise values at each time step, would require $\Omega(k)$ time per release. This is optimal if we wish to release the whole vector. However, queries and updates may be sparse, in which case we could hope to optimise computational time so that query time depends on the number of distinct times t at which a vector cell is queried. A canonical example of such a setting is private sketching, recently studied by Epasto, Mao, Medina, Mirrokni, Vassilvitskii, and Zhong [2023], Holland [2025]. Consider as an example Private Count Sketch [Zhao, Qiao, Redberg, Agrawal, Abbadi, and Wang, 2022, Pagh and Thorup, 2022] which approximately represents a sparse vector under single-entry updates and queries, where each update and queries only involve a logarithmic number of values from the sketch. We present what can be seen as a time-efficient implementation of the binary tree mechanism of Dwork et al. [2010] in the Gaussian noise setting that only materialises the noise values that are actually queried.

Related work. Applying continual observation to linear sketches has previously been considered by Epasto et al. [2023], but their technique requires updating every sketch entry in each time step, making it inefficient in settings where sketches are large. Holland [2025] developed a technique in which sketch updates are buffered and carried out in batches when the number of updates matches the sketch size. While this improves time complexity, buffering updates introduces significant error for large sketches. Our result implies that such trade-offs are not necessary.

Our contribution. We show a new general technique for privately maintaining a vector under continual observation without introducing bias. Consider the binary tree mechanism $\mathcal{M}_{BT}$ with Gaussian noise applied to a vector $A \in \mathbb{R}^k$. We show how there exists a mechanism simulating $\mathcal{M}_{BT}$ such that every vector update and access requires constant worst-case time.

Our technique applies broadly to dynamic private data structures and sketches that can be made private using the Gaussian mechanism.

The idea is to sample a Gaussian additive noise value only when a vector entry is accessed, and to store, for each entry, a logarithmic-size data structure that suffices to produce noise values with the right covariance matrix. We consider the case where time is strictly monotone. For which we describe a data structure such that the noise value for step t can be computed in constant time, regardless of how many time steps have passed since the last noise value was released.

Problem definition and notation. For positive integer c, we use $[c]$ to denote the set $\{1, \cdots, c\}$. For a parameter $\Delta_2 > 0$ and integer n, consider a data set $X = \left\{(u_i, x^{(i)}) : i \in [n]\right\}$, where $u_i \in \mathbb{Z}$, $x^{(i)} \in \mathbb{R}^k$ with $\left\|x^{(i)}\right\|_2 \leq \Delta_2$ for all i. Two data sets are neighbouring if and only if they have symmetric difference of at most 1. We consider the private release of $A^{(t)} = \sum_{i\,:\,u_i \leq t} x_i$ for $t = 1, \ldots, T$. This can be thought of as the continual release of a vector A that is incremented by a vector x_i at time u_i (with several updates being possible at each time step). A query at time step t for entry $i \in [k]$ releases $A_i^{(t)}$ plus an additive noise term. Any number of queries may be performed at each time step t, and we seek privacy guarantees that hold without restrictions on queries. It is possible to skip time steps, performing no queries. Last, for a bit vector $v \in \{0,1\}^*$ and $c \leq |v|$ let $\mathrm{rank}_c(v) = \sum_{i=1}^{c} v_i$ be the number of 1s in the first c bits of v and let $\mathrm{select}_i(v) = \min\{c : \mathrm{rank}_c(v) \geq i\}$ be the position of the ith 1 in v. Notation and definitions related to differential privacy, including the Gaussian mechanism, can be found in Appendix A.

Machine model. Our data structure is presented for a variant of the standard Word RAM model with word size at least $\max(\log n, \log T)$. In particular, this allows rank and select queries on machine words to be answered in constant time. To simplify the exposition, we assume that the Word RAM is augmented with the ability to represent and add real-valued Gaussians, and that fully random hash functions are available. These assumptions can be removed by making use of discrete Gaussians [Canonne, Kamath, and Steinke, 2022].

2 Fast Gaussian Mechanism under Continual Observation

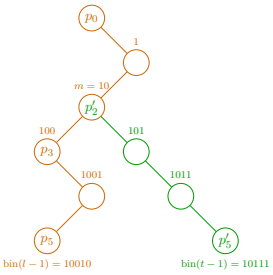
For simplicity, we first consider the case where $k = 1$ such that $A \in \mathbb{R}$. Additionally, we assume that t is monotonically increasing. I.e. no queries to $t' < t$ can be made once time-step t has been reached. Conceptually, if A is queried at every time step, the mechanism corresponds to the classic binary mechanism described in [Dwork et al., 2010], with Gaussian noise replacing Laplace noise. This variant is described and analyzed in Appendix B.

Recall that instead of generating all noisy values for all nodes on the path a priori, we sample the necessary prefix sums on demand. We define an active path as being the path from the root to leaf $\text{bin}(t'-1)$, when A is queried at time t'. To answer queries under differential privacy, we maintain a noise value $\nu^{(t)}$ corresponding to this path. For generating appropriate noise on demand and computing $\nu^{(t)}$ efficiently, we define a structure $DS = \left(l, w^{(t)}, P^{(t)}\right)$ such that $\nu^{(t)} = P^{(t)}_{\log(T)}$ after updating the structure. An update only occurs when A is queried. This structure contains:

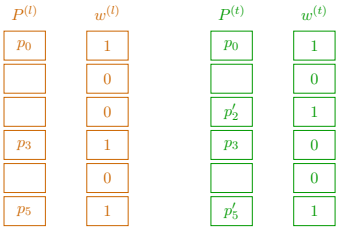
1. the time stamp l of the last time a query was made to A
2. a bit vector $w^{(l)} \in \{0,1\}^{\log(T)}$, that indicates which prefix sums have already been sampled for the active path at time l.
3. a vector $P^{(l)} \in \mathbb{R}^{\log(T)}$ storing either prefix sums of the active path at time l or invalid entries that might contain anything.

For each query at time t, the mechanism returns $A^{(t)} + \nu^{(t)}$, where $\nu^{(t)}$ is updated to match the distribution of the sum of Gaussian variables associated with the dyadic intervals covering t. Each missing value is computed lazily on demand in $\mathcal{O}(1)$ worst-case time. Leaves are labelled with the binary representation of $0, 1 \ldots, T-1$. Internal nodes at level j are labelled by the longest common prefix of their children.

An example where A was last queried at time $l = 19$ and is queried again at time $t = 24$ is shown in Figure 1. In this example, there are 3 prefix sums, which correspond to nodes on the path from the root to $\text{bin}(l-1)$ as indicated by $w^{(t)}$; namely p_0, p_3 and p_5. These are stored in $P^{(l)}$. To update the structure we compute p'_2 by conditioning on its predecessor p_0 and its successor p_3. These can be found efficiently via a predecessor and successor search in $w^{(t)}$ and in doing so we account for the partly shared path and results in $\nu^{(t)} = P^{(t)}_5$. Next, p'_5 is computed by adding an independent sample $\mathcal{N}(0, 3\sigma^2)$ to p'_2. Since we can sample a Gaussian in constant time and that the domain $\log(T)$ can be contained in a machine word, this takes $\mathcal{O}(1)$ worst-case time.



(a) The active path at time l_i, where entry i was queried last, and the new active path at time t. The green part shows where the paths are independent. The longest commen prefix of $\text{bin}(l-1)$ and $\text{bin}(t-1)$ is $m = 10$.



(b) The structure of $w^{(l)}$ and $P^{(l)}$, which corresponds to the orange path in figure (a), and the structure of $w^{(t)}$ and $P^{(t)}$ which is the structure of DS after updating from time l to time t. The latter corresponds to the green path in figure (a). The w vector shows which values correspond to prefix sums in the current active path.

Figure 1: An example of the structure saved for each index $i \in [k]$ where $\log(T) = 6$.

For $0 \leq j \leq \log(T)$ each node $s \in \{0,1\}^j$ corresponds to one dyadic interval of length $T/2^j$. With it we associate an independent Gaussian random variable $\eta_s \sim \mathcal{N}\left(0, \sigma^2\right)$ and a set of time steps that it contains, namely $S(s) = \left\{s \circ 0^{\log(T)-|s|}, \ldots, s \circ 1^{\log(T)-|s|}\right\}$, where $\circ$ denotes concatenation. Let $I_t = \{s \mid t \in S(s)\}$, then the noise at time t should be $\nu_i = \sum_{s \in I_t} \eta_s$. If all η_s were sampled explicitly a priori, computing this sum would take $\Theta(\log(T))$ time. Instead, $\nu^{(t)}$ depends only on

partial sums along the root-to-leaf path of $\text{bin}(l_i - 1)$ that have already been sampled and stored in P. Recall that P_j contains the sum of noise variables for the unique dyadic interval at height $|j|$ containing l_i. When a query is performed at time t, we compute $\nu^{(t)}$ by comparing $\text{bin}(t-1)$ and $\text{bin}(l_i - 1)$. Let m be their longest common prefix of $\text{bin}(l_i - 1)$ and $\text{bin}(t-1)$. We can consider the sum of Gaussian random variables along the path from the root to $\text{bin}(l_i - 1)$ as a random walk. Hence, the missing prefix sum at height $|m|$ can be sampled by conditioning on the previously sampled values above and below (e.g. the successor and predecessor). This is known as a Brownian-bridge step. The remaining path from m to $\text{bin}(t-1)$ is independent of the path to $\text{bin}(l_i - 1)$ and can thus be sampled independently. The bit vector $w^{(t)}$ is then updated to reflect which entries of $P^{(t)}$ correspond to prefix sums on the current active path and which do not. For this structure, we get:

Theorem 2.1 ($\mathcal{O}(1)$ time noise generation for continual counting)**.** *For every positive T, and every $\sigma^2 > 0$, under the assumption that a machine word has size $\Omega(\log(T))$ and rank and select operation on a bit vector of size $O(\log(T))$ are supported in $\mathcal{O}(1)$ worst-case time, there exist a sequence of noise values $\nu^{(1)}, \ldots, \nu^{(T)}$ with the following properties:*

- *For every $0 \leq t < T$, $\nu^{(t)} \sim \mathcal{N}\left(0, \log(T)\sigma^2\right)$ (i.i.d. Gaussian)*
- *Let m be the longest common prefix of bin(t_1-1) and bin(t_2-1) then $\forall\, t_1, t_2$, $Cov(\nu^{(t_1)}, \nu^{(t_2)}) = |m|\sigma^2$.*
- *For every sequence $t_1 < t_2 < t_3 < \ldots$, the sequence of values $\nu^{(t_j)}$, $j = 1, 2, 3, \ldots$ can be computed in $\mathcal{O}(1)$ worst-case time per sample using a data structure DS of $\log(T)$ words.*

We omit the proof of Theorem 2.1 to Appendix C due to space constraints. We can extend the data structure above for $k = 1$ to arbitrary k by running it independently in parallel for each $i \in [k]$. Formally, consider an input sequence of the form $X = \left\{(u_j, x^{(j)}) : j \in [n]\right\}$, where where for all j, $\left\|x^{(j)}\right\|_2 \leq \Delta_2$. Let $\mathcal{M}_{FG}$ be a mechanism that output the vector $A^{(t)} + \nu^{(t)}$ for $t = 1, \ldots, T$, where $A^{(t)} = \sum_{i:u_i \leq t} x_i$ and that runs the structure DS described above independently for each $i \in [k]$. Pseudocode defining the Mechanism $\mathcal{M}_{FG}$, running in worst-case constant time, is given in Appendix C.2. For $\mathcal{M}_{FG}$ we can show:

Lemma 2.2 (Privacy guarantees for $\mathcal{M}_{FG}$)**.** *Two sequences are neighbouring if one can be obtained from the other by adding or removing one vector. $\mathcal{M}_{FG}$ satisfies ρ-zCDP for $\sigma^2 \geq \Delta_2^2/(2\rho)$.*

That the noise values each have the correct entry-wise covariance, the correct distribution, and can be efficiently sampled in worst case $\mathcal{O}(1)$-worst case time via DS_i for each $i \in [k]$ follows from applying the structure from Theorem 2.1 for each i by setting $\sigma^2 \geq \Delta_2^2/(2\rho)$. By this parametrisation of σ^2, Lemma 2.2 guarantees ρ-zCDP. Thus, we can maintain a vector $A \in \mathbb{R}^k$ under continual observation using an additional $\mathcal{O}(\log(T))$ words per $i \in [k]$, no increase in query or update time and without introducing any bias. Namely, each $\nu_i^{(t)}$ can be computed in worst-case time $\mathcal{O}(1)$.

The analysis above depends on the monotonicity of t. A generalisation without this assumption results in a data structure that uses an additional $\mathcal{O}(T)$ words per $i \in [k]$, as the mechanism must store all previously sampled prefix sums from previous time steps. Additionally, because the domain has increased significantly in size (from $\log(T)$ to T), the structure has a worst-case sampling time of $\mathcal{O}(\log\log(T))$ due to successor and predecessor searches. Thus, the generalised version for a non-monotone t would increase the query time by a multiplicative $\mathcal{O}(\log\log(T))$ time. Updating time would still be unaffected.

We conjecture that a similar structure is possible under pure differential privacy, i.e., by constructing a Laplacian bridge.

References

Theodore W. Anderson. *An Introduction to Multivariate Statistical Analysis.* Wiley Series in Probability and Statistics. John Wiley & Sons, Hoboken, NJ, USA, 3 edition, 2003. ISBN 0-471-36091-0, 978-0-471-36091-9.

Joel Daniel Andersson and Rasmus Pagh. A smooth binary mechanism for efficient private continual observation. In *Advances in Neural Information Processing Systems*, volume 36, pages 49133–49145. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/99c41fb9fd53abfdd4a0259560ef1c9d-Paper-Conference.pdf.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. *CoRR*, abs/1605.02065, 2016. doi: 10.48550/arXiv.1605.02065. URL http://arxiv.org/abs/1605.02065.

Clement Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Journal of Privacy and Confidentiality*, 12(1), July 2022. doi: 10.29012/jpc.784. URL https://journalprivacyconfidentiality.org/index.php/jpc/article/view/784.

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3), nov 2011. ISSN 1094-9224. doi: 10.1145/2043621.2043626. URL https://doi.org/10.1145/2043621.2043626. Appeared in Cryptology ePrint Archive 2010/076 and ICALP 2010.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, aug 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. *STOC*, 2010.

Alessandro Epasto, Jieming Mao, Andres Munoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:24, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-263-1. doi: 10.4230/LIPIcs.ITCS.2023.48. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2023.48.

Rayne Holland. Scalable differentially private sketches under continual observation. *CoRR*, abs/2507.03361, 2025. doi: 10.48550/ARXIV.2507.03361. URL https://doi.org/10.48550/arXiv.2507.03361.

Rasmus Pagh and Mikkel Thorup. Improved utility analysis of private countsketch. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 25631–25643. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a47f5cdff1469751597d78e803fc590f-Paper-Conference.pdf.

Sebastiano Vigna. Broadword implementation of rank/select queries. In Catherine C. McGeoch, editor, *Experimental Algorithms*, pages 154–168, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-68552-4.

Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. Differentially private linear sketches: Efficient implementations and applications. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/525338e0d98401a62950bc7c454eb83d-Abstract-Conference.html.

A Omitted Preliminaries

When referring to ρ-zCDP or (zCDP), we refer to the definition by Bun and Steinke [2016]:

Definition A.1 (ρ-Zero-Concentrated Differential Privacy (ρ-zCDP), Bun and Steinke [2016])**.** A randomised mechanism $M : \mathcal{X}^c \to \mathcal{Y}$ is (ε, ρ)-zCDP if for all neighbouring $x, x' \in \mathcal{X}^c$ such that x and x' only differ in one element and for all $\alpha \in (0, \infty)$:

$$D_\alpha(M(x)||M(x')) \leq \varepsilon + \rho\alpha$$

where $D_\alpha(M(x)||M(x'))$ is the α-Rényi divergence between the distribution $M(x)$ and the distribution $M(x')$.
A randomised mechanism $M : \mathcal{X}^c \to \mathcal{Y}$ is ρ-zCDP if it is $(0, \rho)$-zCDP.

When referring to approximately differential privacy and pure differential privacy, we refer to the definition by Dwork and Roth [2014]:

Definition A.2 (Differential Privacy, Dwork and Roth [2014])**.** A randomised mechanism $\mathcal{M} : \mathcal{X}^c \to \mathcal{Y}$ is (ε, δ)-differentially private for $\varepsilon > 0$ and $\delta \geq 0$ if for all subsets of outputs $\mathcal{S} \subseteq \mathcal{Y}$ and for all neighbouring $x, x' \in \mathcal{X}^c$ such that $\|x - x'\|_1 \leq 1$ it holds that:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp\varepsilon \Pr[\mathcal{M}(x') \in \mathcal{S}] + \delta$$

$\mathcal{M}$ satisfy approximate differentially privacy when $\delta > 0$ and pure differential privacy when $\delta = 0$.

Moreover, we characterize the Gaussian Mechanism under zCDP:

Definition A.3 (Gaussian Mechanism, Bun and Steinke [2016])**.** A function $q : \mathcal{X}^n \to \mathbb{R}$ has sensitivity Δ_2 if $\forall x, x' \in \mathcal{X}^n$ differing in a single entry satisfy that $|q(x) - q(x')| \leq \Delta_2$. Let q be a sensitivity-Δ_2 query then the mechanism $\mathcal{M} : \mathcal{X}^n \to \mathbb{R}$, that releases a sample from $\mathcal{N}\left(q(x), \sigma^2\right)$ satisfies $\left(\Delta_2^2/(2\sigma^2)\right)$-zCDP.

B A Classical Continual Counting Mechanism with Gaussian Noise

We revisit a classical mechanism of Dwork, Naor, Pitassi, and Rothblum [2010] for privately releasing estimates of prefix sums $p_t = \sum_{i \leq t} x_i$ of an input stream $x_0, \ldots, x_{T-1} \in \mathbb{R}$, where $T > 1$ is a power of two. Inputs $x_0, \ldots, x_{T-1}$ and $x'_0, \ldots, x'_{T-1}$ are considered neighbouring if they differ in at most one coordinate i where $|x_i - x'_i| \leq 1$. The mechanism is defined in terms of a complete, rooted binary tree with T leaves numbered $0, \ldots, T-1$. We associate a noise variable N_v with each node v of the tree, and let P_v denote the set of nodes on the path from the root node to v. At step $t+1$, after receiving input x_t, the mechanism releases the prefix sum estimate

$$\tilde{p}_t = p_t + \sum_{v \in P_t} N_v \ .$$

It is shown in [Dwork et al., 2010] that if $N_v \sim \mathrm{Lap}((\log_2(T) + 1)/\varepsilon)$ the vector $\tilde{p}$ of prefix sum estimates satisfies ε-differential privacy. In turn, this implies that it satisfies ρ-zero-concentrated differential privacy with $\rho = \varepsilon^2/2$ [Bun and Steinke, 2016]. The variance of each estimate is $\mathrm{Var}[\tilde{p}_t] = 2(\log_2(T) + 1)^3/\varepsilon^2 = (\log_2(T) + 1)^3/\rho$.

Changing the noise distribution. We now consider the same mechanism with Gaussian noise, i.e., where $N_v \sim \mathcal{N}(0, \sigma^2)$, under ρ-zero-concentrated differential privacy. It is clear that $\mathrm{Var}[\tilde{p}_t] = (\log_2(T)+1)\sigma^2$, so what remains is to figure out how σ should depend on ρ and T. It turns out that the maximum variance of this mechanism is better than of other classical binary tree mechanisms such as the mechanism of Chan, Shi, and Song [2011], and it matches the asymptotic error of the much more complicated smooth binary mechanism Andersson and Pagh [2023] .

Let L be the linear map that takes a vector y of dimension $2T-1$ indexed by the vertices in the complete binary tree such that for $t = 0, \ldots, T-1$ we have $(Ly)_t = \sum_{v \in P_t} y_v$. As above we assign numerical indices $v = 0, \ldots, T-1$ to the leaves and use indices $v = T, \ldots, 2T-1$ for the internal nodes. Next, we describe a linear map R such that the composition LR is the all-ones lower triangular matrix, i.e., $LRx = p$. Define the *sign* $s(v)$ of a non-root node v to be -1 if v is a left child and $+1$ if v is a right child, and let $\mathrm{sib}(v)$ denote the sibling of a non-root node v.

$$(Rx)_v = \begin{cases} \frac{1}{2} \sum_{t=0}^{T-1} x_t \text{ if } v \text{ is the root node} \\ \frac{1}{2}\, s(v) \sum_{t:\mathrm{sib}(v) \in P_t} x_t \text{ if } v \text{ is a non-root internal node} \\ \frac{1}{2}\left(x_v - s(v)\, x_{\mathrm{sib}(v)}\right) \text{ if } v \text{ is a leaf} \end{cases} .$$

That is, the root node is associated with the sum of all inputs multiplied by $\frac{1}{2}$, each internal node v is associated with the sum of all inputs corresponding to leaves in its *sibling's* subtree multiplied by $\frac{1}{2}\, s(v)$, and each leaf node is associated with $\frac{1}{2}$ times either a sum or a difference of sibling values. By definition $(LRx)_t = (L(Rx))_t = \sum_{v \in P_t} (Rx)_v$. We argue by induction on the depth that for every internal node w,

$$\sum_{v \in P_w} (Rx)_v = \left(\sum_{t=0}^{\max\{i : w \in P_i\}} x_t \right) - \left(\tfrac{1}{2} \sum_{t : w \in P_t} x_t \right) . \tag{1}$$

That is, the sum along a path from the root to w is equal to the sum of all elements up to and including the leaves below w minus $\frac{1}{2}$ times the sum of leaves below w. This is clearly true when w is the root element. For the induction step we assume that the statement is true for w's parent and see that adding $(Rx)_w$ exactly matches the claimed difference. In turn, for a leaf node t with parent w we have, using (1),

$$\begin{aligned} \sum_{v \in P_t} (Rx)_v &= \left(\sum_{v \in P_w} (Rx)_v \right) + \tfrac{1}{2}\left(x_t - x_{\mathrm{sib(t)}}\right) \\ &= \left(\sum_{i=0}^{\max\{t : w \in P_t\}} x_i \right) - \left(\tfrac{1}{2} \sum_{i : w \in P_i} x_i \right) + \tfrac{1}{2}\left(x_t - s(v)\, x_{\mathrm{sib(t)}}\right) = p_t, \end{aligned}$$

where the last equality is obtained by checking the cases where t is a left and a right child, respectively. With the linear maps L and R defined we see that the mechanism can be written in the form of a factorization mechanism:

$$\tilde{p} = L(Rx + N),$$

where $N \sim \mathcal{N}(0, \sigma^2)^{2T-1}$. To analyze the privacy we first note that the ℓ_2-sensitivity of Rx is $\Delta_2(R) = \frac{1}{2}\sqrt{\log_2(T)+2}$. This is because changing one input x_i by $\Delta \leq 1$ changes exactly $(\log_2(T)+2)$ values $(Rx)_v$ by $\Delta/2 \leq 1/2$, one changed node for each internal level and two at the leaf level.

We can now apply the standard zCDP bound for the Gaussian mechanism to Rx. Releasing $Rx + N$ with $N \sim \mathcal{N}(0, \sigma^2)^{2T-1}$ satisfies ρ-zCDP with $\rho = \Delta_2(R)^2/(2\sigma^2) = (\log_2(T)+2)/(8\sigma^2)$, and since $\tilde{p} = L(Rx + N)$ is a (deterministic) post-processing of $Rx + N$, the released vector $\tilde{p}$ satisfies the same privacy guarantee [Bun and Steinke, 2016]. Equivalently, we can choose $\sigma^2 = (\log_2(T) + 2)/(8\rho)$, resulting in $\mathrm{Var}[\tilde{p}_t] = (\log_2(T) + 1)\sigma^2 < (\log_2(T) + 2)^2/(8\rho)$. This matches the asymptotic variance of the smooth Gaussian mechanism but avoids a lower-order term that is significant whenever T is not very large.

C Omitted Proofs

C.1 Proof of Theorem 2.1

Proof. Let $\mathrm{DS} = \left(l, w^{(l)}, P^{(l)}\right)$ as defined above in section 2. First, consider the space complexity of updating the structure necessary to answer $\nu^{(t)}$ over T time steps. In the binary tree mechanism in Dwork et al. [2010], based on dyadic decompositions, each time step t is contained in exactly one dyadic interval for each of the $\log(T)$ levels of the tree (see the modified version in Appendix B). Hence, at most $\log(T)$ independent Gaussian random variables contribute to ensure privacy for index i over T time steps; each corresponding to a node from the root to the given time step. Additionally, since the global time counter t increases monotonically whenever A is updated, once the mechanism completely passes a dyadic interval, that interval will never be part of a root-to-leaf path for any future time steps. Thus, it suffices to store the prefix sum only along the current path from the root to $\mathrm{bin}(t-1)$ of length $\mathcal{O}(\log(T))$. As a result, $P^{(t)}$ uses at most $\mathcal{O}(\log(T))$ words, and $w^{(t)}$ uses a single machine word of size $\mathcal{O}(\log(T))$. Thus, each entry of $\nu_i^{(t)}$ uses $\mathcal{O}(\log(T)))$ words.

We will briefly outline the structure of $w^{(l)}$, which records which prefix sum in $P^{(l)}$ currently corresponds to the active path from the root to node l. Formally:

$$w_j^{(l)} = \begin{cases} 1 & \text{if } P^{(l)} \text{ has been sampled and corresponds to the path from the root to node } \mathrm{bin}(l-1) \\ 0 & \text{otherwise} \end{cases}$$

As a result, $w^{(l)}$ encodes exactly which prefix sums, corresponding to the root-to-$\mathrm{bin}(l-1)$ path, are sampled. In particular, if $w_j^{(l)} = 1$, the stored value $P_j^{(l)}$ is valid and should be considered in future updates, and if $w_j^{(l)} = 0$, $P_j^{(l)}$ is either not sampled or corresponds to a previous path and should thus be disregarded. Thus, importantly, values in $P^{(t)}$ are never deleted. Instead, when sampled in time t, $w^{(l_i)}$ is updated to $w^{(t)}$ such that only entries consistent with the path to $\mathrm{bin}(t-1)$ remain marked as valid.

Next, consider the time required to sample the noise value $\nu_i^{(t)}$. Namely how we can construct $\left(t, w^{(t)}, P^{(t)}\right)$ from $\left(l, w^{(l)}, P^{(l)}\right)$. Recall that at time step t, $DS = \left(l, w^{(l)}, P^{(l)}\right)$ is available and that: $\nu_i^{(t)} = P_{\log(T)}^{(t)} = \sum_{s \in I_t} \eta_s$, where I_t is the $\mathcal{O}(\log(T))$ dyadic intervals that contains t. Similarly, let I_l be the $\log(T)$ dyadic intervals that contain l. Let m be the largest common prefix of $\mathrm{bin}(l_i - 1)$ and $\mathrm{bin}(t-1)$. Note that m describes the last shared node on their common path and $I_l \cap I_t = I_m$. To compute $P_{\log(T)}^{(t)}$, we need to sample $P_{|m|}^{(t)}$. To do so, we need efficient access to the previously computed successor and predecessor in $P^{(l)}$ for which we will use $w^{(l)}$. Recall that the size of $w^{(l)}$ is $\mathcal{O}(\log(T))$ bits and thus fits in a single machine word by assumption. Vigna [2008] shows that all predecessor and successor queries can be performed in $\mathcal{O}(\log(T))$ worst-case time using rank/

select operations. Specifically for m:

$$\text{succ}(|m|) = \min\left(j > |m| \mid w[j] = 1\right) = \text{select}\left(\text{rank}(|m|) + 1\right)$$
$$\text{pred}(|m|) = \max\left(j < |m| \mid w[j] = 1\right) = \text{select}\left(\text{rank}(|m|) - 1\right)$$

If $l = t$, then the correct prefix sum has already been computed, and we can simply output $P^{(t)}_{\log(T)}$. Otherwise, we can have one of two cases:

1. if l is not set (i.e. index i is queried for the first time)
2. the path from the root to the node $\text{bin}(l-1)$ overlaps with the path to $\text{bin}(t-1)$

For the first case, $w^{(l)}_j = 0\ \forall j \in \{1, \ldots, \log(T)\}$ as no prefix sums have previously been sampled. The path to $\text{bin}(t-1)$ is thus fully independent of previous sampled paths and we can sample $P^{(t)}_{\log(T)} \sim \mathcal{N}\left(0, \log(T)\sigma^2\right)$, set $w^{(t)}_{\log(T)} = w^{(t)}_1 = 1$ and set $l = t$ all in $\mathcal{O}(1)$ worst-case time.

For the second case, $\text{bin}(t-1)$ and $\text{bin}(l-1)$ have a non-empty longest common prefix. Let $a \le m \le b$ be the nearest nodes above and below m such that $w^{(l)}_{|a|} = w^{(l)}_{|b|}$. That is $|a| = \text{pred}(|m|)$ and $|b| = \text{succ}(|m|)$. Recall that these can be found in $\mathcal{O}(1)$-time in $w^{(l)}$ via Vigna [2008]. Per induction, $P^{(l)}_{|a|}$ and $P^{(l)}_{|a|}$ are already generated and correspond to prefix sums along the path from the root to $\text{bin}(l)$. Each of these prefix sums can be considered as a Gaussian random walk. Thus the joint distribution $\left(P^{(l)}_{|a|}, P^{(l)}_{|m|}, P^{(l)}_{|b|}\right)$ is a multivariate Gaussian with covariance $Cov(P^{(l)}_{j_1}, P^{(l)}_{j_2}) = \min\{j_1, j_2\}\sigma^2$ for $(j_1, j_2) \in \{(|a|,|b|), (|b|,|m|), (|a|,|m|)\}$, which correspond to their shared path. Note that a multivariate Gaussian is fully defined by its covariance matrix and its mean (see Anderson [2003]). By conditioning on the endpoints, we get a Brownian bridge. Hence, we can sample $P^{(l)}_{|m|} \mid (P^{(l)}_{|a|}, P^{(l)}_{|b|})$ as:

$$P^{(l)}_{|m|} \mid (P^{(l)}_{|a|}, P^{(l)}_{|b|}) \sim \mathcal{N}\left(P^{(l)}_{|a|} + \frac{|m| - |a|}{|b| - |a|}\left(P^{(l)}_{|b|} - P^{(l)}_{|a|}\right), \frac{(|m| - |a|)(|b| - |m|)}{|b| - |a|}\sigma^2\right)$$

Sampling and setting $P^{(l)}_{|m|}$ accordingly takes $\mathcal{O}(1)$ worst-case time. Once $P^{(l)}_{|m|}$ is sampled, the path below m is independent of all previous samples and of all predecessors of m. Therefore, it can be sampled independently and $P^{(t)}_{\log(T)}$ can be sampled as: Therefore:

$$P^{(t)}_{\log(T)} = P^{(l)}_{|m|} + \mathcal{N}\left(0, (\log(T) - |m|)\sigma_2\right)$$

Note that $P^{(t)}_{\log(T)} \sim \mathcal{N}\left(0, \log(T)\sigma^2\right)$, so overwrites the old value $P^{(l)}_{\log(T)}$ and $P^{(t)}_j = P^{(l)}_j\ \forall j \in (1, \ldots, h_m)$ are valid prefix sums. The remaining are not, as all bits below h_m correspond to nodes in the old subtree containing $\text{bin}(l_i - 1)$, but not $\text{bin}(t-1)$. Hence, the last step is to update $w^{(l)}$ accordingly. Recall that $w^{(l)}$ fits in a single machine word, and thus this can be achieved by clearing the $|m| - 1$ least significant bits from $w^{(l)}$ to obtain $w^{(t)}$. This can be done via bit-shifting in $\mathcal{O}(1)$ worst-case time. Lastly set $w^{(l)}_{\log(T)} = 1$ and update l to t; all in $\mathcal{O}(1)$ worst-case time.

This completes the construction for sampling $\nu^{(t)}_i$ in $\mathcal{O}(1)$ worst-case time. □

C.2 Pseudocode

Algorithm 1 NoiseUpdate(DS_i, t, σ^2, T)

Require: Data structure for coordinate i at last query: $DS_i = (l_i, w^{(i,l_i)}, P^{(i)})$
Ensure: Given $t > l_i$, updated $DS_i' = (l_i', w^{(i,t)}, P^{(i)})$ and noise value $\nu_i^{(t)} = P_L^{(i)}$.

Invariant: If $w_j^{(i,t)} = 1$ then $P_j^{(i)} = P_j^{(i,t)}$.
$L \leftarrow \log T$
if l_i is **unset then** ▷ first query to coordinate i
    $w^{(i,t)} \leftarrow 0$
    $P_0^{(i)} \leftarrow 0$
    Sample $P_L^{(i)} \sim \mathcal{N}(0,\, L\sigma^2)$
    $w_0^{(i,t)} \leftarrow 1$, $w_L^{(i,t)} \leftarrow 1$, ▷ invariant preserved since $P_0^{(i)}$ and $P_L^{(i)}$ have been set
    $l_i' \leftarrow t$
    return $(DS_i' = (l_i', w^{(i,t)}, P^{(i)}),\ \nu_i^{(t)} = P_L^{(i)})$
end if
if $l_i = t$ **then** ▷ already queried at this time
    return $(DS_i,\ \nu_i^{(t)} = P_L^{(i)})$
end if
$u \leftarrow \mathrm{bin}(l_i - 1), \quad v \leftarrow \mathrm{bin}(t-1)$
$m \leftarrow \mathrm{LCP}(u, v)$ ▷ longest common prefix
$h_m \leftarrow |m|$ ▷ height/level where paths split
$h_a \leftarrow \mathrm{pred}_{w^{(i,l_i)}}(h_m), \quad h_b \leftarrow \mathrm{succ}_{w^{(i,l_i)}}(h_m)$ ▷ nearest valid levels above/below h_m
$\mu \leftarrow P_{h_a}^{(i)} + \frac{h_m - h_a}{h_b - h_a}\big(P_{h_b}^{(i)} - P_{h_a}^{(i)}\big)$
$\tau^2 \leftarrow \frac{(h_m - h_a)(h_b - h_m)}{h_b - h_a}\,\sigma^2$
Sample $P_{h_m}^{(i)} \sim \mathcal{N}(\mu,\, \tau^2)$ ▷ Brownian-bridge step conditioning on endpoints
Sample $P_L^{(i)} \sim \mathcal{N}(P_{h_m}^{(i)}, (L - h_m)\sigma^2)$ ▷ sample new independent tail from level h_m to leaf $\mathrm{bin}(t-1)$
$w^{(i,t)} \leftarrow w^{(i,l_i)}$

for $j = h_m + 1$ **to** $L - 1$ **do** ▷ can be implemented in $O(1)$ time with a bit-mask
    $w_j^{(i,t)} \leftarrow 0$
end for ▷ invariant preserved since $P_{h_m}^{(i)}$ and $P_L^{(i)}$ have been set
$l_i' \leftarrow t$
return $(DS_i' = (l_i', w^{(i,t)}, P^{(i)}),\ \nu_i^{(t)} = P_L^{(i)})$