Securing Unbounded Differential Privacy Against Timing Attacks

Zachary Ratliff* Salil Vadhan†

June 2025

Abstract

Recent works [BDDNT23, RV24] have started to theoretically investigate how we can protect differentially private programs against timing attacks, by making the joint distribution the output and the runtime differentially private (JOT-DP). However, the existing approaches to JOT-DP have some limitations, particularly in the setting of unbounded DP (which protects the size of the dataset and applies to arbitrarily large datasets). First, the known conversion of pure DP programs to pure JOT-DP programs in the unbounded setting [BDDNT23] (a) incurs a constant additive increase in error probability (and thus does not provide vanishing error as $n \to \infty$) (b) produces JOT-DP programs that fail to preserve the computational efficiency of the original pure DP program and (c) is analyzed in a toy computational model in which the runtime is defined to be the number of coin flips. For approximate JOT-DP, an efficient conversion with vanishing error in the RAM model is known [HPN11, RV24], but only applies to programs that run in $O(n)$ time on datasets of size n, as linear runtime is implied by "timing stability," the timing analogue of global sensitivity. In this work, we overcome these limitations. Specifically:

1. We show that the error required for pure JOT-DP in the unbounded setting depends on the model of computation.
   - In a randomized RAM model where the dataset size n is given (or can be computed in constant time) and we can generate random numbers (not just random bits) in constant time, polynomially small error probability is necessary and sufficient.
   - If n is not given or we only have a random-bit generator, an (arbitrarily small) constant error probability is necessary and sufficient.
2. The aforementioned positive results are proven by efficient procedures to convert any pure JOT-DP program P in the upper-bounded setting to a pure JOT-DP program P' in the unbounded setting, such that the output distribution of P' is γ-close in total variation distance to that of P, where γ is either an arbitrarily small constant or polynomially small, depending on the model of computation.

*Harvard University & OpenDP. Email: zacharyratliff@g.harvard.edu. Supported in part by Cooperative Agreement CB20ADR0160001 with the Census Bureau, and in part by Salil Vadhan's Simons Investigator Award

†Harvard University & OpenDP. Email: salil_vadhan@harvard.edu

Contents

1 Introduction

Timing side-channel attacks, in which an attacker measures the runtime of an algorithm to infer otherwise private information, have proven to be a highly effective method for breaking modern cryptographic systems. These attacks have been used to extract signing keys from cryptographic processors [Koc96], recover private keys from remote TLS servers [BB05, BT11, AFP13, AP16], and leak sensitive information from hidden hardware states (e.g., Spectre [KHF+20] and Meltdown [LSG+18]).

Early research on timing attacks focused on identifying and mitigating timing side-channels in cryptographic implementations. However, the growing adoption of *differential privacy* (DP) [DMNS06]—an influential framework for privacy-preserving data analysis, used in applications such as analyzing user behavior [MDH+20, G+16] and releasing aggregate statistics [Abo18, ABC+20]—has introduced new avenues for timing attacks. Differential privacy achieves its protections by adding carefully calibrated noise to computations, ensuring that the output of a query is not too sensitive to the addition or removal of a single data point. However, researchers have already identified that the runtime of user-defined queries [HPN11, AKM+15] and noise samplers [JMRO21] can severely violate the promises of differential privacy. Thus, recent research has begun to theoretically investigate *timing-private* DP programs, which ensure that the joint distribution of the output and runtime satisfies differential privacy [BDDNT23, RV24].

Definition 1.1 ((ε, δ)-Joint Output/Timing Privacy [BDDNT23, RV24])**.** Let $\mathcal{X}$ be a dataset space with an adjacency relation, $\mathcal{E}$ be a set of execution environments for a computational model, $\mathcal{Y}$ be an output space, $\mathcal{T} \subseteq \mathbb{R}^{\geq 0}$ a set of possible runtimes in the model, and $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ a randomized program in the model. Then we say that P is (ε, δ)-*jointly output/timing-private* (JOT-DP) if for all adjacent $x, x' \in \mathcal{X}$, all pairs of input-compatible execution environments $\texttt{env}, \texttt{env}' \in \mathcal{E}$, and all $S \subseteq \mathcal{Y} \times \mathcal{T}$

$$\Pr[(Y, T) \in S] \leq e^{\varepsilon} \cdot \Pr[(Y', T') \in S] + \delta$$

where $Y = \texttt{out}(P(x, \texttt{env}))$ and $T_P(x, \texttt{env})$ denote the output and runtime of P respectively (and defined similarly for Y' and T'). When $\delta = 0$, we say that P achieves *pure* JOT-DP, and when $\delta > 0$, we say that P achieves *approximate* JOT-DP. The dataset space $\mathcal{X}$ and its adjacency relation determine whether the program achieves JOT-DP in the *bounded* (the number of records $n = |x|$ in a dataset $x \in \mathcal{X}$ is fixed and publicly known), *unbounded* ($|x|$ can be arbitrarily large and is intended to remain private), or *upper-bounded* (upper bound on $|x|$ is publicly known, but $|x|$ should remain private subject to the upper bound) settings.

The above notion of timing privacy requires that the joint distribution of the program P's output and runtime when executed in the environment $\texttt{env}$ satisfies the standard definition of differential privacy. Informally, the execution environment represents the

state of the computer before execution, for example, the contents of memory. We give further background on execution environments in Section 2.

Beyond its role in data analysis, DP has recently emerged as a valuable tool for designing more practical and efficient cryptographic systems. For instance, metadata-private messaging systems have leveraged weaker DP guarantees to improve performance over purely cryptographic approaches [LGZ18, VDHLZZ15]. Similarly, the concept of *differential obliviousness*, which extends DP-like guarantees to the memory access patterns of a process, has been proposed as a lightweight alternative to oblivious RAM [CCMS22]. Apple has even adopted DP in their implementation of *enhanced visual search* as a more efficient method of anonymizing user queries [App24, ABG+24]. Thus, there is a pressing need to understand the extent to which DP implementations can be protected from timing attacks.

1.1 Limitations of Prior Approaches

One general technique suggested by Haeberlen, Pierce, and Narayan [HPN11] and formalized and generalized by Ratliff and Vadhan [RV24], for constructing JOT-DP programs involve first establishing a bound on the program's *timing stability*, which is the timing equivalent of global sensitivity. Once this bound is established, the output release is delayed by an amount sampled from a distribution, scaled according to the timing stability and the desired privacy level. However, this technique seems to inherently result in *approximate* privacy ($\delta > 0$) because it can only add non-negative "noise" to the program's runtime. It is possible to achieve *pure* ($\delta = 0$) JOT-DP in the bounded-DP (or upper-bounded DP) setting, where the maximum size of the input is known and public, by padding execution to the worst case runtime. However, in the *unbounded* DP setting, where the size of the program's input is unknown and must be protected by differential privacy, achieving JOT-DP for non-trivial programs introduces several interesting challenges.

First, Ben Dov, David, Naor, and Tzalik [BDDNT23] show that pure DP programs in the unbounded setting can be converted into JOT-DP programs, but the conversion results in a constant additive increase in error probability (and thus does not provide vanishing error as the dataset size $n \to \infty$). It remains unknown whether one must tolerate this constant probability of constant additive error, or if it is possible to construct pure JOT-DP programs that maintain the same asymptotic accuracy guarantees as their non-timing-private versions (which typically have error probability that tends to 0 as the dataset size n grows). Furthermore, the conversion technique of [BDDNT23] can produce in a JOT-DP program that is inefficient, even if the original pure DP program is efficient. The reason being that constructing the sampler for the JOT-DP program requires computing the probabilities of all possible outputs of the original program, which may take exponential time even if the original program runs in expected polynomial time.

For approximate JOT-DP, the previously mentioned technique of adding random delays can achieve vanishing error, but it is only applicable to programs that are timing stable,

which implies an $O(n)$ runtime on datasets of size n. Consequently, in the unbounded DP setting, this approach cannot ensure timing privacy for common DP algorithms with superlinear runtimes. For example, the *smooth-sensitivity median* algorithm [NRS07] for differentially private median computation requires sorting the input data, which incurs a $\Omega(n \log n)$ runtime for any comparison-based sorting algorithm. Thus, in the unbounded DP setting, no method is currently known for constructing even *approximately* JOT-DP analogues of DP algorithms that require superlinear runtime.

1.2 Our Results

In this work, we address the above limitations and introduce new techniques for constructing differentially private programs in the unbounded setting that are secure against timing attacks. We adopt the RAM model of computation with a random number generator, which we view as more natural than the model used by Ben Dov et al. [BDDNT23] (see Section 2). Their results used a toy computational model equipped solely with a random bit generator, where the total runtime was measured by the number of random bits generated before halting. Additionally, since we focus on the unbounded setting of differential privacy, we do not utilize the Word RAM model, as its finite word size[1] imposes constraints on addressable memory and, consequently, on input size. However, we note that our transformations do not rely on the RAM model's capability to generate integers that are super-polynomially large in the program's runtime.

Within the RAM model of computation, we answer an open question by Ben Dov et al. [BDDNT23] and construct an efficient procedure for converting any pure DP program P in the *upper-bounded* setting (see Definition 1.1) to a pure JOT-DP program P' for the unbounded setting whose output distribution differs by at most a constant β in total variation distance from P.

Theorem 3.4 (Pure JOT-DP RAM programs in the Unbounded Setting)**.** For all $\beta > 0$, $\varepsilon > 0$, $\varepsilon' > \varepsilon$, and ε-JOT-DP RAM programs $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ in the *upper*-bounded setting, there exists a ε'-JOT-DP RAM program $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ for the unbounded setting such that

$$\left\| \mathtt{out}(P(x, \mathtt{env})) - \mathtt{out}(P'(x, \mathtt{env})) \right\|_{TV} < \beta$$

Furthermore, the mechanism $P'(x, \mathtt{env})$ is simply an explicit algorithm that makes one oracle call to P on a truncation of x along with an additional computation that takes time $O(|x|)$ with high probability.

Given the above result, we can convert pure JOT-DP programs with superlinear runtime in the upper-bounded setting into pure JOT-DP programs in the unbounded setting. Note

[1] The Word RAM model sometimes defines the word size ω as $\omega = O(\log n)$, where n is the program's input length. However, this introduces additional complexities, as the word size itself may reveal information about the input length.

that the requirement that P has a pure JOT-DP implementation in the upper-bounded setting is not overly restrictive, as many useful programs have such implementations (for example, by padding all executions to the maximum runtime on datasets of size within the upper bound). This resolves the open problem of Ben Dov et al. [BDDNT23].

Note that Theorem 3.4 allows for a constant additive error in the output with constant probability, even as $n \to \infty$. This raises the question of whether such an error is unavoidable, a question we address in this work. Specifically, we show that the answer depends on the computational model starting with the following lower bound:

Theorem 4.1 (Lower Bound for Pure JOT-DP Programs)**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ be a ε-JOT-DP program in the unbounded setting for some computational model, and that releases an estimate of its true input length. Let $t_0 = O(1)$ be the expected runtime of P on the empty dataset, and $p_{t_0}(n) \leq 1$ be the smallest non-zero probability such that $Z \sim Bernoulli(p_{t_0}(n))$ can be sampled in time $2 \cdot t_0$ when P is executed on a dataset consisting of n records equal to a fixed value (e.g., 0). Then $\exists c$ such that $\forall x \in \mathcal{X}, |x| \geq c$:

$$\Pr\left[\left|\texttt{out}(P(x, \texttt{env})) - n\right| > n - c\right] > p_{t_0}(n)$$

for $n = |x|$ as $n \to \infty$.

The above theorem gives a *computational-model-dependent* lower bound on the achievable utility for pure JOT-DP programs. In particular, with probability at least $p_{t_0}(n)$, the program will output a useless result (since the error is of the order $O(n)$). If our computational model includes only a coin-tossing function as in the BDNNT model (Section 2), then $p(n) \geq 2^{-2 \cdot t_0}$, so the error probability is at least a constant. On the other hand, we show that in a randomized RAM model of computation where the input length is given as part of the input, we can achieve error that decays inverse polynomially in the input length (Section 4).

Theorem 4.5 (Pure JOT-DP RAM Programs in the Unbounded Setting)**.** For all $\varepsilon > 0$, $\varepsilon' > \varepsilon$, and ε-DP RAM programs $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ in the upper-bounded setting, there exists a ε'-JOT-DP RAM program $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ for the unbounded setting satisfying $\forall x \in \mathcal{X}$ of length n, $\forall$ input-compatible $\texttt{env} \in \mathcal{E}$, $\forall c \geq 2$:

$$\left\|\texttt{out}(P(x, \texttt{env})) - \texttt{out}(P'(x, \texttt{env}))\right\|_{TV} < O\left(\frac{1}{n^c}\right)$$

Furthermore, the mechanism $P'(x, \texttt{env})$ is an explicit algorithm that makes one oracle call to P on a truncation of x along with an additional computation that takes time $O(|x|)$ with high probability.

In particular, we obtain the following pure JOT-DP implementation of the Laplace mechanism.

Corollary 4.6 (Pure JOT-DP Laplace Mechanism)**.** For all $\varepsilon > 0$, and all $c > 0$, there exists an ε-JOT-DP RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ for releasing sums in the unbounded setting such that for all datasets $x \in \mathcal{X}$:

$$\Pr\left[\left|\texttt{out}(P(x, \texttt{env})) - \sum x_i\right| \geq \frac{C \cdot \ln(n)}{\varepsilon}\right] < O\left(\frac{1}{n^c}\right)$$

where $n = |x|$ is the size of the dataset, and $C > 0$ is a universal constant.

Theorem 4.5 shows that we can construct *pure* JOT-DP sums in the randomized RAM model that achieve comparable privacy guarantees to the standard ε-DP Laplace mechanism. While the error probability is not exponentially decaying as in the standard (non-timing-private) Laplace mechanism, the JOT-DP mechanism still ensures that the error vanishes at a polynomial rate. This is an improvement over the constant error bound given by Ben Dov et al. [BDDNT23] and Theorem 3.4, and is the best possible by Theorem 4.1 since $p_{t_0}(n) \geq n^{-2^{2 \cdot t_0}}$ (Section 4).

1.3 Techniques

We introduce a new technique for constructing *pure* JOT-DP programs in the unbounded setting (Section 3). Specifically, both Theorem 3.4 and Theorem 4.5 are obtained by (a) constructing a pure JOT-DP program that coarsely estimates its input length, (b) truncating the dataset based on this estimate, and (c) running an upper-bounded pure JOT-DP program using the estimate as the upper bound on dataset size.

Informally, the coarse estimation procedure repeatedly queries whether the input length exceeds some threshold, where the threshold doubles in each round, continuing until the answer is "no." Because the query responses must satisfy differential privacy, each has an associated error probability. However, this error can be made arbitrarily small.

To achieve the bound for RAM programs in Theorem 4.5, we leverage the fact that the input length n is known in advance, allowing the error parameter to explicitly depend on n. Specifically, the algorithm uses an adaptive sampling procedure that repeatedly flips a biased coin, starting with an initial success probability of $1/(n + k)^c$ for constants c and k. After each unsuccessful flip, the success probability is updated to $1/(n - i + k)^c$ until it reaches a fixed probability of $1/k^c$. Flipping continues until the first success, at which point the algorithm outputs the total number of flips. Importantly, this ensures (a) the runtime is exactly determined by the output, and (b) the output itself is an ε-DP estimate of the input length, making the algorithm ε-JOT-DP. Using this adaptive sampler as a building block, we construct pure JOT-DP RAM programs that achieve the bound from Theorem 4.5. Our construction is optimal in terms of its failure probability (i.e., it achieves error probability that vanishes at a polynomial rate). In particular, Theorem 4.1 establishes that this decay rate is the best achievable within the RAM model, and in fact, weaker computational models can only achieve a constant error probability.

To establish the lower bound in Theorem 4.1, we analyze the output support of a pure DP program in the unbounded setting conditioned on halting within time $t \leq 2t_0$, where t_0 is the expected runtime of the program on the empty input. Our argument relies on the observation that even when the database is very long, the mechanism must with some probability behave exactly as it does on the empty database. On the empty database the mechanism halts within time $2 \cdot t_0$ with high probability, and thus the same behavior must occur with positive probability on the long database. By the definition of JOT-DP, the output support conditioned on halting within this time bound must be finite and identical across all inputs. We further show that every output in this finite support provides poor utility for most inputs. Nonetheless, the mechanism must output an element from this finite set with probability at least $p_{t_0}(n)$, where $p_{t_0}(n)$ is the smallest non-zero probability computable by a program running in time $2t_0$. Together, our results demonstrate that in many reasonable computational models, pure JOT-DP programs in the unbounded setting necessarily incur an added utility cost to maintain their timing privacy guarantees.

2 Preliminaries

RAM Model. Throughout this work, we will use the idealized RAM model of computation. The RAM model consists of an infinite sequence of memory cells, each capable of storing arbitrarily large natural numbers. Variables are stored in registers and RAM programs can perform a set of basic operations for arithmetic (addition, subtraction, multiplication, and integer division), Boolean logic (AND, OR, NOT), and reading/writing memory. They also allow conditional jumps (e.g., **if** CONDITIONAL **goto** LINE), which implement standard control flow constructs such as **if** and **if/else** statements. Additionally, we allow our RAM programs to use randomness by executing a $\texttt{RAND}(n)$ instruction to uniformly sample an integer from $\{0, \ldots, n\}$. The runtime of a RAM program is defined as the number of basic instructions[2] executed before the program halts, meaning that the set of possible runtime values $\mathcal{T}$ corresponds to the natural numbers $\mathbb{N}$. We also include a built-in variable `input_len` and `input_ptr` (respectively `output_len` and `output_ptr`), which stores the length of the program's input and its location in memory respectively (and similarly for the program's output).

We also consider a variant of the randomized RAM model where the program lacks direct access to the input length (i.e., `input_len` is not initialized). This model is strictly weaker than the one described above, as the program must compute the input length during execution if needed. In this model, the end of the input is marked in memory by a special delimiter symbol reserved for indicating the end of the input.

$\text{RAM}_{\text{BDDNT}}$. Earlier work by Ben Dov et al.[BDDNT23] used a computational model

[2]We count conditional branching instructions and randomness sampling as basic instructions taking one time step.

where randomness is provided by a simple coin-tossing function (i.e., a draw from Bernoulli(1/2)), and runtime is measured solely by the number of coin tosses performed. This is in contrast to our RAM model, where a call to `RAND`(n) samples a random integer from $\{0, \ldots, n\}$ and runtime is measured by the total number of instructions executed before halting. Consequently, when discussing the main results of Ben Dov et al. in Section 3, we denote by $\text{RAM}_{\text{BDDNT}}$ the RAM model in which randomness is restricted to calls to a simple Bernoulli(1/2) sampler and runtime is measured by the total number of such calls.

Execution Environments. The runtime of a program on a given input is often highly dependent on the program's *execution environment*. For example, a program executing on x86 hardware will have runtime that can be influenced by concurrent processes executing on the system, the state of the branch predictor (if the hardware supports speculative execution), the cache state, and various other forms of resource contention that might occur on the system. Thus, we always consider a program's runtime to be a function of both the input and some execution environment $\texttt{env} \in \mathcal{E}$ representing the machines initial state before executing the program. For example, the set of execution environments $\mathcal{E}$ for RAM programs includes all possible memory configurations and initial values of built-in variables.

Definition 2.1 (RAM Execution Environment)**.** The *execution environment* `env` of a RAM program is the infinite sequence $(v_0, v_1, \ldots,)$ such that $M[i] = v_i$ for all i along with the values stored by the built-in variables such as `input_ptr` and `input_len`, `output_ptr`, and `output_len`.

We note that while the execution environment significantly influences program runtimes in practice, throughout this paper, we will work with RAM programs whose runtime and output distributions are jointly independent of the execution environment. Additionally, we emphasize that not every execution environment is *compatible* with a given input. For example, a RAM program that processes a length-n string consisting entirely of 1's may be incompatible with a RAM environment where memory has been zeroed out. We say that $P(x, \texttt{env})$ is undefined for such environments `env` that are incompatible with an input x. Consequently, our definitions are often quantified over all pairs of inputs $x \in \mathcal{X}$ and *input-compatible* execution environments $\texttt{env} \in \mathcal{E}$.

The Bounded, Upper-Bounded, and Unbounded Settings. Differential privacy is defined with respect to an input space $\mathcal{X}$, where inputs typically consist of $n \geq 0$ records drawn from some domain $\mathcal{D}$. Formally, we define the input space as $\mathcal{X} = \bigcup_{n=0}^{\infty} \mathcal{D}^n$. Two inputs $x, x' \in \mathcal{X}$ are said to be *adjacent* with respect to a dataset distance metric $d_{\mathcal{X}}$ if $d_{\mathcal{X}}(x, x') \leq 1$.

We can distinguish between different settings of differential privacy according to whether the program's input size is considered public information. In the *bounded* setting, the input length is assumed to be publicly known. Consequently, privacy guarantees do not extend

to hiding the dataset size. This assumption simplifies the defense against timing attacks, as execution can often be padded to match the worst-case runtime for inputs of a given length. A stronger flavor of DP is the *upper-bounded* setting, where only an upper bound $n_{\max}$ on the program's input length is assumed to be public. This setting is commonly encountered in practice, as one can typically establish a conservative upper limit on the dataset size. Similarly to the bounded setting, one can often pad execution up to a worst case runtime[3] in the upper-bounded setting to protect against timing attacks. However, the strongest privacy guarantee is provided by the *unbounded* setting, in which the input length is intended to be kept private and can be arbitrarily large.

In both the upper-bounded and unbounded settings, adjacency is often defined using an *insert-delete* distance.

Definition 2.2 (Insert-Delete Distance)**.** For $x \in \mathcal{D}^*$, an insertion to x is an addition of an element z to a location in x resulting in a new input $x' = [x_1, \ldots, x_i, z, x_{i+1}, \ldots, x_n]$. Likewise, a deletion from x is the removal of an element from a location i, giving a new input $x' = [x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. We define the *insert-delete distance*, denoted d_{ID}, of inputs $x, x' \in \mathcal{D}^*$ to be the minimum number of insertion and deletion operations needed to transform x into x'.

Throughout this paper, we operate within the unbounded setting of differential privacy and will therefore implicitly use d_{ID} as the adjacency relation in all definitions and theorems.

Properties of JOT-DP Programs. We will frequently use the fact that DP programs with constant-time execution on all inputs are also JOT-DP.

Lemma 2.3 (Constant-Time JOT-DP Programs [RV24])**.** If a program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ is ε-DP in its output and there exists a constant c such that $T_P(x, \texttt{env}) = c$ for all $x \in \mathcal{X}$ and $\texttt{env} \in \mathcal{E}$, then P is ε-JOT-DP.

We will also make use of programs P that are the composition of JOT-DP programs P_1 and P_2. Such programs P are constructed by *chaining* together P_1 such that its output along with its unaltered input are fed as the input to program P_2.

Lemma 2.4 (Sequential Composition of JOT-DP RAM Programs [RV24])**.** Let $P_1 : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be an ε_1-JOT-DP RAM program. Let $P_2 : \mathcal{X} \times \mathcal{E} \to \mathcal{Z} \times \mathcal{E}$ be an ε_2-JOT-DP RAM program. Then the sequentially composed program $P_2 \otimes P_1 : \mathcal{X} \times \mathcal{E} \to (\mathcal{Y} \times \mathcal{Z}) \times \mathcal{E}$ that executes P_1 on the input x followed by P_2 on input x is $(\varepsilon_1 + \varepsilon_2)$-JOT-DP.

Finally, we review the Discrete Laplace distribution.

[3]However, in practice, this approach may be inefficient, as the program's worst-case runtime for inputs of length $n_{\max}$ could be prohibitively long.

Definition 2.5 (Discrete Laplace Distribution)**.** The *Discrete Laplace distribution* with shift $\mu \in \mathbb{N}$ and scale $s > 0$, is supplied a $x \in \mathbb{Z}$, has the probability mass function:

$$p(x \mid \mu, s) = \frac{e^{1/s} - 1}{e^{1/s} + 1} \cdot e^{-|x-\mu|/s}$$

The cumulative distribution function (CDF) is given by:

$$F(x \mid \mu, s) = \begin{cases} \frac{e^{1/s}}{e^{1/s}+1} \cdot e^{-(\mu - x)/s}, & \text{if } x \leq \mu, \\ 1 - \frac{1}{e^{1/s}+1} \cdot e^{-(x-\mu)/s}, & \text{if } x > \mu. \end{cases}$$

Throughout this paper, we will often use a *censored*[4] version of the Discrete Laplace distribution, which we denote by $\texttt{CensoredDiscreteLaplace}(\mu, s, \ell, u)$. This distribution is additionally parameterized by lower and upper bounds ℓ and u so that $X \sim \text{DiscreteLaplace}(\mu, s)$ is clamped to the range $[\ell, u]$.

Lemma 2.6 (Censored Discrete Laplace is DP [GRS12])**.** Let x, x' be adjacent datasets, $\ell, u \in \mathbb{N}$ for $\ell \leq u$, $f : \mathcal{X} \to \mathbb{Z}$ a function with global sensitivity Δ, and $\varepsilon > 0$. Then $M(x) = \texttt{CensoredDiscreteLaplace}(\mu = f(x), s = \Delta/\varepsilon, \ell, u)$ is ε-DP.

We remark that constant-time instantiations of the censored Discrete Laplace distribution are known [BV19, RV24]. Using Lemma 2.6 and the fact that we can implement the censored Discrete Laplace mechanism to run in fixed time that depends only on s, ℓ, and u, we obtain a JOT-DP version of `CensoredDiscreteLaplace` (by Lemma 2.3).

Lemma 2.7 (Censored Discrete Laplace is JOT-DP)**.** Let x, x' be adjacent datasets, $\ell, u \in \mathbb{N}$ for $\ell \leq u$, $f : \mathcal{X} \to \mathbb{Z}$ a function with global sensitivity Δ, and $\varepsilon > 0$. Then there exists a RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ such that $P(x, \texttt{env})$ samples from $\texttt{CensoredDiscreteLaplace}(\mu = f(x), s = \Delta/\varepsilon, \ell, u)$ in time $O(u)$ and achieves ε-JOT-DP.

3 Pure Timing-Private Programs

In this section, we characterize pure JOT-DP programs in the unbounded setting. To begin, we reintroduce a result from Ben Dov et al. [BDDNT23] for the $\text{RAM}_{\text{BDDNT}}$ model of computation (described in Section 2).

Theorem 3.1 (Ben Dov et al. [BDDNT23])**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be any ε-DP $\text{RAM}_{\text{BDDNT}}$ program. For any $\beta > 0$, $\varepsilon' > \varepsilon$ there exists a ε'-JOT-DP $\text{RAM}_{\text{BDDNT}}$ program $P' : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ such that for all $x \in \mathcal{X}$, $\texttt{env} \in \mathcal{E}$:

$$\left\| \texttt{out}(P(x, \texttt{env})) - \texttt{out}(P'(x, \texttt{env})) \right\|_{TV} < \beta$$

[4]The literature also refers to this as the Truncated Geometric Mechanism [GRS12].

Program 1 General Construction for Pure JOT-DP Programs in the Unbounded Setting

Input: A dataset x, privacy parameter $\varepsilon' > 0$, failure parameter $\beta > 0$, and ε-JOT-DP program P for the upper-bounded setting.

Output: The output of P when executed on the input x truncated to $\hat{n}$ records where $\hat{n}$ is a ε'-DP count on the size of x.

$\varepsilon' = \varepsilon'/2$;
$\beta' = \beta/2$;
$m = \frac{2}{\varepsilon'} \cdot \lceil \ln(\frac{1}{\beta'}) \rceil$;
while True do
  Scan first m rows of input and set $\mu = \min\{n, m\}$; {note that $n = |x|$}
  $s = 1/\varepsilon'$;
  $\ell = 0$; {lower bound for censored Discrete Laplace mechanism}
  $u = m$; {upper bound for censored Discrete Laplace mechanism}
  $\hat{n} =$ `CensoredDiscreteLaplace`(μ, s, ℓ, u);
  if $\hat{n} < \frac{m}{2}$ **then**
    $s = 1/\varepsilon$;
    $\hat{x} =$ `Truncate`(x, m); {Truncates the dataset to m records if $|x| > m$ }
    run $P^m(\hat{x})$; {P in the m-upperbounded setting}
    return `output of` $P(\hat{x})$;
  else
    $\varepsilon' = \varepsilon'/2$;
    $\beta' = \beta'/2$;
    $m = \frac{2}{\varepsilon'} \cdot \lceil \ln(\frac{1}{\beta'}) \rceil$;

Theorem 3.1 is promising in that it suggests one can achieve pure JOT-DP as long as one is willing to tolerate a constant error probability. However, the authors left as an open question whether the pure JOT-DP program P' can maintain the computational efficiency of P. We give a positive answer to this question for a wide class of programs (namely, programs that have pure JOT-DP implementations in the upper-bounded setting).

We start with the construction described by Program 1, which first computes a pure JOT-DP *over*-estimate of a program's input length that holds with arbitrarily small probability β. Intuitively, Program 1 scans its input m_i entries at a time, where m_i depends only on the parameters ε', β, and i, and m_{i+1} is roughly $2m_i$. In each iteration, the program releases a DP count on the first m_i entries of the input (i.e., a DP count on m_i if $m_i < |x|$, and a DP count on $|x|$ otherwise) and stops scanning the input once it releases a count less than $m_i/2$. This step consumes ε' of the privacy budget to generate the coarse estimate m_k, which serves as an upper bound on the input size with high probability. Once this estimate is determined, the input is truncated (if necessary) in time that depends only on

the differentially private overestimate m_k. Finally, the program P^m is executed on the transformed dataset $\hat{x}$, where P^m is the program P restricted to inputs of length at most m_k.

Lemma 3.2. Let $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ be the RAM program implementing Program 1 for a RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that is ε-JOT-DP in the upper-bounded setting. Then P' achieves $(\varepsilon + \varepsilon')$-JOT-DP in the unbounded setting.

Proof. We can interpret the program P' as the composition of two subprograms: $P_{\texttt{(line 13)}}$, which encompasses the execution of P' up to line 13, and P^m, which represents the execution of P on $\hat{x}$ within the upper-bounded setting, where inputs are guaranteed to have length at most m. By assumption, P^m satisfies ε-JOT-DP. Furthermore, `Truncate` is 1-stable[5], and therefore for all $x, x' \in \mathcal{X}$ satisfying $d_{\mathrm{ID}}(x, x') \leq 1$, it follows that $d_{\mathrm{ID}}(\texttt{Truncate}(x, m_k), \texttt{Truncate}(x', m_k)) \leq 1$. Thus, to establish the overall privacy guarantee, it remains to show that m is computed in a differentially private manner and that $P_{\texttt{(line 13)}}$ satisfies ε'-JOT-DP. Given these conditions, we can view the execution of $P'(x, \texttt{env})$ as the sequential composition of two JOT-DP programs $P_{\texttt{(line 13)}}(x, \texttt{env})$ and $P^m(\hat{x}, \hat{\texttt{env}})$.

Observe that $P_{\texttt{(line 13)}}$ scans the input m entries at a time, where m depends only on ε' and β' (lines 3 and 18). At each iteration i of the loop (for $i = 1 \ldots k$), let $\varepsilon_i = \varepsilon'$, $\beta_i = \beta'$, and $m_i = m$. During each loop iteration, the program invokes a censored Discrete Laplace mechanism to compute a ε_i-DP count over the first $m_i = 2 \cdot \lceil \ln(1/\beta_i) \rceil / \varepsilon_i$ entries of the input where $\varepsilon_i = \varepsilon'/2^i$ and $\beta_i = \beta/2^i$. Recall that each invocation of the censored Discrete Laplace mechanism is ε_i-JOT-DP (Lemma 2.7). Thus, up to line 11, the runtime of Program 1 depends only on the number of iterations k, which is a post-processing function of the k JOT-DP counts. Furthermore, once the condition on line 10 is met, the program performs a truncation operation, $\texttt{Truncate}(x, m)$. This operation executes in time dependent only on m_k (e.g., by returning a copy of the first m_k records of x). Since $P'_{\texttt{(line 13)}}$ computes k ε_i-DP counts (for $i = 1, \ldots, k$) and runs in time determined solely by k, it follows that $P'_{\texttt{(line 13)}}$ satisfies ε'-JOT-DP since

[5] A function $f : \mathcal{X} \to \mathcal{X}$ that maps datasets to datasets is 1-stable if for all $x, x' \in \mathcal{X}$, $d_{\mathrm{ID}}(f(x), f(x')) \leq c \cdot d_{\mathrm{ID}}(x, x')$.

$$\begin{aligned}\sum_{i=1}^{k} \varepsilon_i &\leq \sum_{i=1}^{\infty} \varepsilon_i \\ &= \sum_{i=1}^{\infty} \frac{\varepsilon'}{2^i} \\ &= \varepsilon' \cdot \sum_{i=1}^{\infty} \frac{1}{2^i} \\ &= \varepsilon'\end{aligned}$$

Consequently, P' is the composition of two JOT-DP programs, $P'_{\texttt{(line 13)}}$ and P. By the basic composition theorem (Lemma 2.4), it follows that the overall program satisfies $(\varepsilon + \varepsilon')$-JOT-DP. □

Lemma 3.3. Let $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ be the program implementing Program 1 for a program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that is ε-JOT-DP in the upper-bounded setting. Then P' returns the output of P on dataset x with probability at least $1 - \beta$.

Proof. We first consider the case where Program 1 terminates on loop iteration k with $m_k < |x|$. In this scenario, $\hat{x} = \texttt{Truncate}(x, m_k)$ and $|\hat{x}| < |x|$ so the program returns the output of P on the first m_k entries of x. The probability of this occurring is given by

$$\Pr\left[\hat{n}_i < \frac{m_i}{2}\right] \leq e^{-m_i \cdot \varepsilon_i / 2} \leq \beta_i$$

for $\hat{n}_i \sim \texttt{CensoredDiscreteLaplace}(\mu = m_i, s = 1/\varepsilon_i, \ell = 0, u = m_i)$ by the CDF of the Discrete Laplace distribution and the fact that $m_i \geq \frac{2}{\varepsilon_i} \cdot \ln(1/\beta_i)$. We union bound over the k invocations of the mechanism:

$$\begin{aligned}\sum_{i=1}^{k} \Pr\left[\hat{n}_i < \frac{m_i}{2}\right] &\leq \sum_{i=1}^{\infty} \beta_i \\ &= \sum_{i=1}^{\infty} \frac{\beta}{2^i} \\ &= \beta \sum_{i=1}^{\infty} \frac{1}{2^i} \\ &= \beta\end{aligned}$$

We now condition on Program 1 halting on loop iteration k where $m_k \geq |x|$. In this case, $\hat{x} = x$ since $\texttt{Truncate}(x, m_k)$ does not alter x when $m_k > |x|$. Thus P' returns the output of P executed on x and the claim follows. □

We now prove that for every pure JOT-DP program in the upper-bounded setting, we can obtain a pure JOT-DP program in the unbounded setting.

Theorem 3.4 (Pure JOT-DP in the Unbounded Setting)**.** For all $0 < \beta < 1$, $\varepsilon > 0$, $\varepsilon' > \varepsilon$, and ε-JOT-DP RAM programs $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ in the *upper*-bounded setting, there exists a ε'-JOT-DP RAM program $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ for the unbounded setting such that

$$\Big\| \mathtt{out}(P(x, \mathtt{env})) - \mathtt{out}(P'(x, \mathtt{env})) \Big\|_{TV} < \beta$$

Furthermore, the mechanism $P'(x, \mathtt{env})$ is simply an explicit algorithm that makes one oracle call to P on a truncation of x along with an additional computation that takes time $O(|x|)$ with probability at least $1 - \beta - e^{-\Omega(|x|)}$.

Proof. The proof follows from Lemma 3.2 and Lemma 3.3. In particular, given P, we construct the program P' as described in Program 1 so that P' is ε'-JOT-DP for $\varepsilon' > \varepsilon$. What remains to be shown is that, with high probability, P' runs in time $O(|x|)$ before making the oracle call to P.

There are two cases. We first consider the case where $m_k < |x|$. As shown in Lemma 3.3, this happens with probability:

$$\begin{aligned}
\sum_{i=1}^{\infty} \Pr\left[\hat{n}_i < \frac{m_i}{2}\right] &\leq \sum_{i=1}^{\infty} e^{-m_i \cdot \varepsilon_i / 2} \\
&\leq \sum_{i=1}^{\infty} \beta_i \\
&= \beta
\end{aligned}$$

where $\hat{n}_i \sim \mathtt{CensoredDiscreteLaplace}(\mu = m_i, s = 1/\varepsilon_i, \ell = 0, u = m_i)$. Thus, we consider the case where $m_k > |x|$ and show that P' stops looping when $m_k = O(|x|)$ with high probability. First, observe that

$$\begin{aligned}
\frac{m_k}{m_{k-1}} &= \frac{\frac{2^k}{\varepsilon'} \cdot \left\lceil \log \frac{2^k}{\beta} \right\rceil}{\frac{2^{k-1}}{\varepsilon'} \cdot \left\lceil \log \frac{2^{k-1}}{\beta} \right\rceil} \\
&= 2 \cdot \frac{r+1}{r}
\end{aligned}$$

for $r = k - 1 + \left\lceil \log(1/\beta) \right\rceil \geq 1$. Thus,

$$2 \leq \frac{m_k}{m_{k-1}} \leq 4$$

Therefore, during each iteration of the loop, we increase the number of input entries scanned by at least a factor of 2 and at most a factor of 4. Now, consider the index k such that $m_k > |x|$ but $m_{k-1} \leq |x|$. Then $|x| < m_k \leq 4 \cdot |x|$. Furthermore, there exists a $j \geq k$ such that $4 \cdot |x| \leq m_j \leq 16 \cdot |x|$. Note that during this loop iteration we have $\hat{n}_j \sim \texttt{CensoredDiscreteLaplace}(\mu = |x|, s = 1/\varepsilon_j, \ell = 0, u = m_j)$ and

$$\Pr\left[\hat{n}_j > \frac{m_j}{2}\right] = 1 - F\left(\frac{m_j}{2} | \mu = |x|, s = 1/\varepsilon_j\right) \leq e^{-\Omega(|x|)}$$

where F is the CDF of the Discrete Laplace distribution. Thus, with probability at least $1 - \beta - e^{-O(|x|)}$, P' stops looping on iteration $j = O(\log |x|)$ where $m_j = O(|x|)$. The runtime of each loop iteration is dominated by the time it takes to sample from $\texttt{CensoredDiscreteLaplace}(\mu, s, \ell, u = m_i)$, which can be bounded by $O(m_i)$ (Lemma 2.7), where u starts at m_1 and at least doubles during each iteration up to $m_j = O(|x|)$. It follows that, with high probability, P' runs in time $O(|x|)$ before making the oracle call to P and terminating. □

Theorem 3.4 shows that for any DP RAM program P, it is possible to efficiently construct a JOT-DP variant P' with comparable privacy guarantees, incurring at most a β increase in error probability. In other words, the modified program $P'(x)$ may return inaccurate results with probability at most β higher than $P(x)$. However, the construction used in Theorem 3.4 does not extend to the $\text{RAM}_{\text{BDDNT}}$ model, as it relies on sampling from a Censored Discrete Laplace distribution using a constant number of coin flips. This sampling is possible only if all output probabilities are dyadic, but Lemma A.2 shows that no nontrivial Censored Discrete Laplace distribution has this property. Consequently, even in the bounded DP setting, one cannot implement the standard Laplace mechanism with *perfect* timing privacy guarantees, as can be done in the RAM model. Fortunately, we demonstrate that replacing the sampling from the Censored Discrete Laplace distribution in line 9 of Program 1 with sampling from a closely related distribution whose probability masses are entirely dyadic ensures that Theorem 3.4 also holds in the $\text{RAM}_{\text{BDDNT}}$ model (see Appendix).

Whether the constant additive increase in error probability β is unavoidable even in the RAM model remains an open question. In the next section, we establish a lower bound on the achievable utility of pure JOT-DP programs in the unbounded setting.

4 A Lower Bound for JOT-DP Programs

A key question that arises from Theorem 3.4 (Section 3) is whether the constant additive error in the output, which occurs with constant probability even as the input size n grows,

is unavoidable. In this section, we address this question, exploring the inherent limitations of JOT-DP programs. We begin by establishing a lower bound, demonstrating that the achievable utility for pure JOT-DP programs depends on the computational model.

Theorem 4.1 (Lower Bound for JOT-DP Counting Programs)**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ be a ε-JOT-DP program in the unbounded setting for some computational model, and that releases an estimate of its true input length. Let $t_0 = O(1)$ be the expected runtime of P on the empty dataset, and $p_{t_0}(n) \leq 1$ be the smallest non-zero probability such that $Z \sim \mathit{Bernoulli}(p_{t_0}(n))$ can be sampled in time $2 \cdot t_0$ when P is executed on a dataset consisting of n records equal to a fixed value (e.g., 0). Then $\exists c$ such that $\forall x \in \mathcal{X}, |x| \geq c$:

$$\Pr\left[\left|\texttt{out}(P(x, \texttt{env})) - n\right| > n - c\right] > p_{t_0}(n)$$

for $n = |x|$ as $n \to \infty$.

Proof. Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a ε-JOT-DP program for releasing an estimate on the length of its input. Let $\mathbb{E}[T_P(\lambda, \texttt{env}_\lambda)] = t_0$ where $t_0 = O(1)$ and λ is the empty dataset and $\texttt{env}_\lambda$ is an input-compatible execution environment. Let $p_{t_0}(n) \leq 1$ be the smallest probability such that $Z \sim \mathit{Bernoulli}(p_{t_0}(n))$ is sampleable in time $2 \cdot t_0$ when P is executed on a dataset consisting of n copies of a fixed record.

Observe that $\Pr[T_P(\lambda, \texttt{env}_\lambda) < 2 \cdot t_0] \geq 1/2$ by Markov's inequality. Furthermore, let

$$S = \text{supp}(\texttt{out}(P(\lambda, \texttt{env}_\lambda)) | T_P(\lambda, \texttt{env}_\lambda) \leq 2 \cdot t_0)$$

Note that S is finite since $P(\lambda, \texttt{env}_\lambda)$ can only generate a finite number of outputs within $2 \cdot t_0$ time steps. We can set $c = \max_{y \in S} y$. Then $\forall n \geq c, y \in S$ it follows that

$$|y - n| \geq n - c$$

By pure JOT-DP, we have that for all x

$$\text{supp}(\texttt{out}(P(x, \texttt{env})) | T_P(x, \texttt{env}) \leq 2 \cdot t_0) = S$$

Thus, we have $\Pr\left[|\texttt{out}(P(x, \texttt{env})) - n| > n - c\right] > p_{t_0}(n)$ where $n = |x|$. □

Theorem 4.1 establishes that the achievable utility of pure JOT-DP programs in the unbounded setting depends on the computational model. For example, in the randomized RAM model, as we have defined it, where the program receives its input length n as part of the input, then $p_{t_0}(n) \geq n^{-2^{2 \cdot t_0}}$ implying that $p_{t_0}(n)$ vanishes as $n \to \infty$. This can be realized, for instance, by a RAM program that applies repeated squaring to the input length and then invokes a random number generator on the result. Conversely, if the RAM program *does not* receive its input length n as part of its input (e.g., as described in Section 2), the best achievable bound[6] is $p_{t_0}(n) \geq 2^{-O(2^{2t_0})}$, which only guarantees a constant failure probability. In the next section, we present explicit constructions of pure JOT-DP RAM programs in the unbounded setting where $p(n)$ vanishes as $n \to \infty$.

[6]Obtainable by starting with a constant and performing repeated squaring.

4.1 Pure JOT-DP RAM Programs in the Unbounded Setting

We now present a general construction for converting a JOT-DP RAM program P in the upper-bounded setting into a JOT-DP RAM program P' in the unbounded setting. Notably, P' achieves an error probability that matches the best achievable bound (up to constant factors) within the RAM model, as established by the lower bound in Theorem 4.1. Intuitively, our construction follows a similar approach to Program 1, but leverages the fact that the program has access to the input length n. The program begins by computing a coarse estimate of the dataset size, ensuring with high probability that this estimate serves as a valid upper bound on the input length $|x|$. Given this upper bound, we then apply the same technique as in Program 1, truncating the dataset if necessary before executing the upper-bounded JOT-DP program P.

Program 2 takes as input a dataset $x \in \mathcal{D}^*$ and outputs an estimate of the dataset size using an adaptive coin-flipping process. The program initializes a biased coin with a success probability of $1/(n+k)^c$ for constants c and k. It then repeatedly flips the coin until the first success, recording the total number of flips as the output. On the $(i+1)$th flip, the success probability is adjusted to $1/(n-i+k)^c$, and this adjustment continues until the success probability reaches a fixed value of $1/k^c$, at which point the bias is no longer updated. Thus, the PMF of the output of Program 2 is given below.

$$f_{(n,c)}(i) = \begin{cases} \frac{1}{(n-i+k)^c} \cdot \prod_{j=0}^{x-1} \left(1 - \frac{1}{(n-j+k)^c}\right) & \text{if } 0 \le i \le n, \\ \mathrm{Geom}_{\frac{1}{k^c}}(i-n) \cdot \prod_{j=0}^{n-1} \left(1 - \frac{1}{(n-j+k)^c}\right) & \text{if } i > n \end{cases}$$

where $\mathrm{Geom}_p(i)$ is the PMF of the Geometric distribution defined as:

$$\mathrm{Geom}_p(i) = \begin{cases} (1-p)^{i-1} \cdot p & \text{if } i \in \{1, 2, 3, \dots\}, \\ 0 & \text{otherwise.} \end{cases}$$

Importantly, the runtime of Program 2 is fully determined by its output. Thus, if the program's output satisfies ε-DP, it is also true that the program will satisfy ε-JOT-DP.

Lemma 4.2. For all $c \ge 2$, $k \ge 2$, the RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ described in Program 2 is ε-JOT-DP in the unbounded setting where $\varepsilon = 2c \cdot \ln\left(\frac{k+1}{k-1}\right)$.

Proof. Let $p_n(j) = 1/(n-j+k)^c$ and observe that

$$p_{n+1}(j+1) = \frac{1}{(n+1-j-1+k)^c} = \frac{1}{(n-j+k)^c} = p_n(j)$$

and therefore for all $y < n$

$$\prod_{j=0}^{y} \left(\frac{1-p_n(j)}{1-p_{n+1}(j)}\right) = \frac{1-p_n(y)}{1-p_{n+1}(0)} \le \frac{(n+1+k)^c}{(n+1+k)^c - 1} \le \frac{k^c}{k^c - 1}$$

Program 2 A RAM Program for Approximate Timing-Private Counts

Input: A dataset x occupying memory locations $M[0], \ldots, M[\texttt{input\_len} - 1]$. The values $c, k \geq 2$ are hardcoded constants.

Output: A noisy estimate of the input length $|x|$.

```
1: n = input_len;
2: count = 0;
3: flag = 0;
4: while flag == 0 do
5:    v = n − count; {Note the RAM model rounds negative numbers to 0}
6:    b = v + k;
7:    B = 1;
8:    for j = 0, . . . , (c − 1) do
9:       B = B · b; {Computing b^c}
10:   r = RAND(B);
11:   if r == 0 then
12:      flag = 1;
13:   else
14:      count = count + 1;
15:
16: return count;
```

and similarly

$$\prod_{j=0}^{y} \left(\frac{1 - p_{n+1}(j)}{1 - p_n(j)} \right) = \frac{1 - p_{n+1}(0)}{1 - p_n(y)} \leq \frac{(n - y + k)^c}{(n - y + k)^c - 1} \leq \frac{k^c}{k^c - 1}$$

We now consider the ratio of PMFs.

Case 1 ($y \leq n$):

$$\begin{aligned}
\frac{f_n(y)}{f_{n+1}(y)} &= \frac{p_n(y)}{p_{n+1}(y)} \cdot \prod_{j=0}^{y-1} \left(\frac{1-p_n(j)}{1-p_{n+1}(j)} \right) \\
&= \frac{p_n(y)}{p_{n+1}(y)} \cdot \frac{1-p_n(y-1)}{1-p_{n+1}(0)} \\
&\leq \left(\frac{(n+1-y+k)^c}{(n-y+k)^c} \right) \cdot \frac{k^c}{k^c-1} \\
&\leq \frac{(k+1)^c}{k^c} \cdot \frac{k^c}{k^c-1} \\
&\leq \frac{(k+1)^c}{k^c-1} \\
&\leq \left(\frac{k+1}{k-1} \right)^{2c}
\end{aligned}$$

Similarly,

$$\begin{aligned}
\frac{f_{n+1}(y)}{f_n(y)} &= \frac{p_{n+1}(y)}{p_n(y)} \cdot \prod_{j=0}^{y-1} \left(\frac{1-p_{n+1}(j)}{1-p_n(j)} \right) \\
&= \frac{p_{n+1}(y)}{p_n(y)} \cdot \left(\frac{1-p_{n+1}(0)}{1-p_n(y-1)} \right) \\
&= \frac{(n-y+k)^c}{(n+1-y+k)^c} \cdot \left(\frac{k^c}{k^c-1} \right) \\
&\leq \frac{(n-y+1+k)^c}{(n-y+1+k)^c-1} \\
&\leq \frac{(k+1)^c}{(k+1)^c-1} \\
&\leq \frac{(k+1)^c}{k^c-1} \\
&\leq \left(\frac{k+1}{k-1} \right)^{2c}
\end{aligned}$$

Case 2 ($y > n$):

$$\begin{aligned}
\frac{f_n(y)}{f_{n+1}(y)} &= \frac{\mathrm{Geom}(p=\frac{1}{k^c}, y-n)}{\mathrm{Geom}(p=\frac{1}{k^c}, y-n-1)} \cdot \left(\frac{1}{1-p_{n+1}(n)}\right) \cdot \prod_{j=0}^{n-1} \left(\frac{1-p_n(j)}{1-p_{n+1}(j)}\right) \\
&= \left(\frac{k^c}{k^c}\right) \cdot \frac{(1-\frac{1}{k^c})^{y-n-1}}{(1-\frac{1}{k^c})^{y-n-2}} \cdot \left(\frac{(k+1)^c}{(k+1)^c-1}\right) \cdot \left(\frac{1-p_n(n-1)}{1-p_{n+1}(0)}\right) \\
&\leq \frac{(k+1)^c}{(k+1)^c-1} \cdot \left(\frac{k^c}{k^c-1}\right) \\
&\leq \left(\frac{k+1}{k-1}\right)^{2c}
\end{aligned}$$

and similarly,

$$\begin{aligned}
\frac{f_{n+1}(y)}{f_n(y)} &= \frac{\mathrm{Geom}(p=\frac{1}{k^c}, y-n-1)}{\mathrm{Geom}(p=\frac{1}{k^c}, y-n)} \cdot (1-p_{n+1}(n)) \cdot \prod_{j=0}^{n-1} \left(\frac{1-p_{n+1}(j)}{1-p_n(j)}\right) \\
&\leq \frac{k^c}{k^c-1} \cdot \left(\frac{1-p_{n+1}(0)}{1-p_n(n-1)}\right) \\
&\leq \frac{k^c}{k^c-1} \cdot \left(\frac{k^c}{k^c-1}\right) \\
&\leq \left(\frac{k+1}{k-1}\right)^{2c}
\end{aligned}$$

It follows that the program is ε-DP for $\varepsilon = 2c \cdot \ln\left(\frac{k+1}{k-1}\right)$. Furthermore, observe that the runtime of this program is a deterministic function of its output. Specifically,

$$T_P(x, \texttt{env}) = 5 + (\texttt{out}(P(x, \texttt{env})) + 1) \cdot (7 + 2(c-1))$$

Since there exists a deterministic function f such that $T_P(x, \texttt{env}) = f(\texttt{out}(P(x, \texttt{env})))$, if $\texttt{out}(P(x, \texttt{env}))$ is ε-DP, then by post-processing, $(\texttt{out}(P(x, \texttt{env})), T_P(x, \texttt{env}))$ must also satisfy ε-DP. Thus, the claim follows. □

We now describe how to set k appropriately. By Lemma 4.2, we have

$$\varepsilon \geq 2c \cdot \ln\left(\frac{k+1}{k-1}\right) = 2c \cdot \ln\left(1 + \frac{2}{k-1}\right)$$

When k is large, we can use the approximation that $\ln(1+x) \approx x$ for small x, and

therefore

$$\begin{aligned}\varepsilon &= 2c \cdot \ln\left(1 + \frac{2}{k-1}\right) \\ &\approx 2c \cdot \frac{2}{k-1} \\ &= \frac{4c}{k-1}\end{aligned}$$

Rearranging the inequality, it suffices to choose $k = O(c/\varepsilon)$ and then run Program 2 to obtain a JOT-DP estimate of the input length of a RAM program. We now demonstrate that this estimate is sufficiently accurate to derive an *upper bound* on the input length, where the probability that the upper bound is less than the true length of the input decays inverse polynomially in n.

Lemma 4.3. For all $c \geq 2, k \geq 2$, the RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ described in Program 2 outputs an estimate $\hat{y} = \texttt{out}(P(x, \texttt{env}))$ of $|x| = n$ that satisfies

$$\Pr\left[\hat{y} < \frac{n}{2}\right] \leq O\Big(\frac{1}{n^{c-1}}\Big)$$

Proof. Let $n = |x|$ and $\hat{y} = \texttt{out}(P(x, \texttt{env}))$ be the output of Program 2. We have that

$$\begin{aligned}\Pr[\hat{y} < \frac{n}{2}] &= \sum_{i=1}^{\frac{n}{2}} \left(p_n(i) \prod_{j=0}^{i-1} (1 - p_n(j)) \right) \\ &\leq \sum_{i=1}^{\frac{n}{2}} p_n(i) \qquad \text{(union bound)} \\ &\leq \frac{n}{2} \cdot p_n\Big(\frac{n}{2}\Big) \\ &= \frac{\frac{n}{2}}{(\frac{n}{2} + k)^c} \\ &\leq O\Big(\frac{1}{n^{c-1}}\Big)\end{aligned}$$

and the claim follows. □

By Lemma 4.3, we can set $m = 2 \cdot \hat{y}$, ensuring that with probability $O(1/n^{c-1})$, we have $m \geq |x|$. We can then apply the same technique as in Section 3 and compose Program 2 with an arbitrary RAM program P that is JOT-DP in the upper-bounded setting. We describe the general construction in Program 3.

Program 3 JOT-DP RAM Program for the Unbounded Setting

Input: A dataset x, and a privacy parameter $\varepsilon_1 > 0$. The program P is hardcoded into line 4

Output: A noisy estimate of the input length $|x|$.

1: Set $\hat{n}$ to the output of Program 2 with constants k and c such that $k = O(c/\varepsilon')$;
2: $\hat{n} = 2 \cdot \hat{n}$;
3: $\hat{x} = \texttt{Truncate}(x, \hat{n})$; {returns the first $\hat{n}$ entries of x if $|x| > \hat{n}$}
4: run ε_2-JOT-DP program $P^{\hat{n}}(\hat{x})$ and return the result {$P^{\hat{n}}$ bounds inputs to length $\hat{n}$}

Lemma 4.4. For all $c \geq 2$, $k \geq 2$, the RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ described in Program 2 runs in time $O(n)$ with probability at least $1 - (1 - \frac{1}{k^c})^n$, where $n = |x|$.

Proof. The runtime of Program 2 is a deterministic function of its output. Specifically,

$$\begin{aligned} T_P(x, \texttt{env}) &= 5 + (\texttt{out}(P(x, \texttt{env})) + 1) \cdot (6 + 2(c-1)) \\ &= O(\texttt{out}(P(x, \texttt{env}))) \end{aligned}$$

Thus, it suffices to show that $\Pr[\texttt{out}(P(x, \texttt{env})) \leq 2n]$ holds with high probability. Observe that:

$$\begin{aligned} \Pr[\texttt{out}(P(x, \texttt{env})) > 2n] &\leq \text{Geom}\Big(p = \frac{1}{k^c}, n\Big) \\ &\leq \Big(1 - \frac{1}{k^c}\Big)^n \end{aligned}$$

□

Theorem 4.5 (Pure JOT-DP RAM Programs)**.** For all $\varepsilon > 0$, $\varepsilon' > \varepsilon$, and ε-DP RAM programs $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ in the upper-bounded setting, there exists a ε'-JOT-DP RAM program $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ for the unbounded setting satisfying $\forall x \in \mathcal{X}$ of length n, $\forall$ input-compatible $\texttt{env} \in \mathcal{E}$, $\forall c \geq 2$:

$$\Big\| \texttt{out}(P(x, \texttt{env})) - \texttt{out}(P'(x, \texttt{env})) \Big\|_{TV} < O\left(\frac{1}{n^c}\right)$$

Furthermore, the mechanism $P'(x, \texttt{env})$ is an explicit algorithm that makes one oracle call to P on a truncation of x along with an additional computation that takes time $O(|x|)$ with high probability.

Proof. The proof follows a similar structure to that of Theorem 3.4. In line 1 of Program 3, a noisy estimate $\hat{n}$ of $n = |x|$ is obtained by executing Program 2. By Lemma 4.2, Program 2 satisfies ε_1-JOT-DP in the unbounded setting. Furthermore, lines 2-3 can be executed in a fixed number of instructions. Thus, we represent lines 1-3 of Program 2 as the ε_1-JOT-DP program $P_{(\texttt{line 3})}$.

Next, we consider the composition of $P_{(\texttt{line 3})}$ with the ε_2-JOT-DP program P. By sequential composition, it follows that Program 2 is ε'-JOT-DP in the unbounded setting, where $\varepsilon' = \varepsilon_1 + \varepsilon_2$. What remains to be shown are the accuracy and runtime guarantees of Program 3.

By Lemma 4.4, we have that, with high probability, Program 2 runs in time $O(n)$ before making a single oracle call to P on a truncated version of x. Observe that, conditioned on $\hat{n} > \frac{n}{2}$ in line 1, we have $\hat{x} = x$ after executing line 3. Thus, Program 3 returns the output of the ε-JOT-DP program P on input x. By Lemma 4.3, Program 2 in line 1 returns an estimate $\hat{n}$ such that

$$\Pr\left[\hat{n} < \frac{n}{2}\right] = O\left(\frac{1}{n^{c-1}}\right)$$

Therefore, after executing line 2 so that $\hat{n} = 2 \cdot \hat{n}$, it follows that $\hat{n} > |x|$ with probability at least $1 - O(\frac{1}{n^{c-1}})$. Conditioning on this case, $\hat{x} = x$ after executing line 3, and therefore Program 2 will return the output of $P^{\hat{n}}$ on input x, where $P^{\hat{n}}$ accepts inputs of length at most $\hat{n}$ and satisfies $\hat{n} > |x|$. □

Corollary 4.6 (Pure JOT-DP Laplace Mechanism)**.** For all $\varepsilon > 0$, and all $c > 0$, there exists an ε-JOT-DP RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ for releasing sums in the unbounded setting such that for all datasets $x \in \mathcal{X}$:

$$\Pr\left[\left|\texttt{out}(P(x, \texttt{env})) - \sum x_i\right| \geq \frac{C \cdot \ln(n)}{\varepsilon}\right] < O\left(\frac{1}{n^c}\right)$$

where $n = |x|$ is the size of the dataset, and $C > 0$ is a universal constant.

Proof. Fix $c \geq 2$. Let M be the Censored Discrete Laplace mechanism that adds noise drawn from a Discrete Laplace distribution and clips the output to lie within the bounds $\ell = 0$ and $u = \Delta \cdot U$, where U is an upper bound on the input length and each $x_i \in [0, \Delta]$. By Lemma 2.7, there exists an ε'-JOT-DP RAM program $P_{\text{Lap}} : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that implements M, and furthermore P_{Lap} is a ε'-JOT-DP RAM program for the upper-bounded setting since it accepts inputs of length at most U. Let Z denote the output of the censored Discrete Laplace program P_{Lap}. Because $Z \sim \texttt{CensoredDiscreteLaplace}(\mu = \sum x_i, s = \Delta/\varepsilon', \ell = 0, u = \Delta \cdot U)$, we have

$$\Pr[|Z - \mu| \geq t] \leq \exp\left(-\frac{\varepsilon' \cdot t}{\Delta}\right)$$

Applying Theorem 4.5 with $\varepsilon > \varepsilon'$ gives an ε-JOT-DP RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ such that, for every x of length n and every input-compatible $\texttt{env}$

$$\left\|\texttt{out}(P_{\mathrm{Lap}}(x, \texttt{env})) - \texttt{out}(P(x, \texttt{env}))\right\|_{TV} = O\big(n^{-c}\big)$$

Thus, by a union bound:

$$\Pr\Big[\big|\texttt{out}(P(x, \texttt{env})) - \sum_{i=1}^{n} x_i\big| \geq t\Big] \leq \exp\left(-\frac{\varepsilon' \cdot t}{\Delta}\right) + O\big(n^{-c}\big)$$

We choose $t = \Delta \cdot c \ln n/\varepsilon'$ so that

$$\Pr\Big[\big|\texttt{out}(P(x, \texttt{env})) - \sum_{i=1}^{n} x_i\big| \geq t\Big] \leq \frac{1}{n^c} + O\big(n^{-c}\big) = O\big(n^{-c}\big)$$

Thus, setting $C = \Delta \cdot c \cdot \varepsilon/\varepsilon'$ and substituting $t = C \ln n/\varepsilon$:

$$\Pr\Big[\big|\texttt{out}(P(x, \texttt{env})) - \sum_{i=1}^{n} x_i\big| \geq \frac{C \ln n}{\varepsilon}\Big] < O\big(n^{-c}\big)$$

□

References

[ABC+20] Ahmet Aktay, Shailesh Bavadekar, Gwen Cossoul, John Davis, Damien Desfontaines, Alex Fabrikant, Evgeniy Gabrilovich, Krishna Gadepalli, Bryant Gipson, Miguel Guevara, et al. Google covid-19 community mobility reports: anonymization process description (version 1.1). *arXiv preprint arXiv:2004.04145*, 2020.

[ABG+24] Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Guy N Rothblum, Kunal Talwar, Karl Tarbe, Ruiyu Zhu, et al. Scalable private search with wally. *arXiv preprint arXiv:2406.06761*, 2024.

[Abo18] John M Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867, 2018.

[AFP13] Nadhem J Al Fardan and Kenneth G Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. In *2013 IEEE symposium on security and privacy*, pages 526–540. IEEE, 2013.

[AKM+15] Marc Andrysco, David Kohlbrenner, Keaton Mowery, Ranjit Jhala, Sorin Lerner, and Hovav Shacham. On subnormal floating point and abnormal timing. In *2015 IEEE Symposium on Security and Privacy*, pages 623–639. IEEE, 2015.

[AP16] Martin R Albrecht and Kenneth G Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of tls. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I 35*, pages 622–643. Springer, 2016.

[App24] Apple. Privacy-preserving machine learning with homomorphic encryption. https://machinelearning.apple.com/research/homomorphic-encryption, 2024. Accessed: 2025-01-19.

[BB05] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[BDDNT23] Yoav Ben Dov, Liron David, Moni Naor, and Elad Tzalik. Resistance to timing attacks for sampling and privacy preserving schemes. In *4th Symposium on Foundations of Responsible Computing (FORC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[BT11] Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *European Symposium on Research in Computer Security*, pages 355–371. Springer, 2011.

[BV19] Victor Balcer and Salil Vadhan. Differential privacy on finite computers. *Journal of Privacy and Confidentiality*, 9:2, 2019.

[CCMS22] T-H Hubert Chan, Kai-Min Chung, Bruce Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. *ACM Journal of the ACM (JACM)*, 69(4):1–49, 2022.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 1995.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[G+16] Andy Greenberg et al. Apple's "differential privacy" is about collecting your data–but not your data. *Wired, June*, 13(1), 2016.

[GRS12] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[HPN11] Andreas Haeberlen, Benjamin C Pierce, and Arjun Narayan. Differential privacy under fire. In *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[JMRO21] Jiankai Jin, Eleanor McMurtry, Benjamin IP Rubinstein, and Olga Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. *arXiv preprint arXiv:2112.05307*, 2021.

[KHF+20] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pages 104–113. Springer, 1996.

[LGZ18] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 711–725, 2018.

[LSG+18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *arXiv preprint arXiv:1801.01207*, 2018.

[MDH+20] Solomon Messing, Christina DeGregorio, Bennett Hillenbrand, Gary King, Saurav Mahanti, Zagreb Mukerjee, Chaya Nayak, Nate Persily, Bogdan State, and Arjun Wilkins. Facebook Privacy-Protected Full URLs Data Set, 2020.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

[RV24] Zachary Ratliff and Salil Vadhan. A framework for differential privacy against timing attacks. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3615–3629, 2024.

[VDHLZZ15] 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.

A Appendix

We show that programs that attempt to avoid *all* information leakage through their runtime (i.e., $T_P(x, \texttt{env}) \equiv T_P(x', \texttt{env}')$ for all $x, x' \in \mathcal{D}^*$ and $\texttt{env}, \texttt{env}' \in \mathcal{E}$) will experience some loss in utility. In particular, we show that programs for computing means will exhibit a constant additive error in their output even as the length of the input $n \to \infty$. This result suggests that some portion of the program's privacy budget must be allocated to privatizing the program's runtime.

Lemma A.1. Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a ε-DP RAM program for computing the mean $\frac{1}{|x|}\sum_{i=1} x_i$ such that $T_P(x, \texttt{env}) \equiv T_P(x', \texttt{env}')$ for all $x, x' \in \{0,1\}^*$ and input-compatible $\texttt{env}, \texttt{env}' \in \mathcal{E}$. Then for every $0 < \beta < 1$, there exists a dataset x and a constant α such that

$$\Pr\left[\left|\texttt{out}(P(x, \texttt{env})) - \frac{1}{|x|}\sum x_i\right| > \alpha\right] > \beta$$

Proof. Let F be the cumulative distribution function of the runtime random variable $T_P(x, \texttt{env})$. We pick $p > 0$ and let $t = F^{-1}(1-p)$. Then $\Pr[T_P(x, \texttt{env}) > t] = p$. When $T_P(x, \texttt{env}) \leq t$, the program can read at most t entries of the input. By lower bounds for samplers [CEG95], for every $0 < \beta < 1$, there exists an input x such that, conditioned on reading at most t locations of x, the algorithm fails to output an α-accurate estimate $\hat{\mu}$ of $\mu = \frac{1}{|x|}\sum x_i$ with probability at least β for

$$\alpha = \Omega\left(\sqrt{\frac{\log(1/\beta)}{t}}\right)$$

Thus, without conditioning, the program fails to output an α-accurate estimate of μ with probability at least $\beta - p$. We take $\beta = 2p$ so that t is a fixed constant and hence so is α. □

Lemma A.2 (No Fully-Dyadic Censored Discrete Laplace)**.** Fix integers $\ell < \mu < u$ and a scale $s > 0$. Let Z be the Discrete Laplace random variable with shift $\mu \in \mathbb{Z}$, censored to

the support $\{\ell, \ell+1, \ldots, \mu, \mu+1, \ldots, u\}$:

$$\Pr[Z = k] = \begin{cases} F(\ell \mid \mu, s) & \text{if } k = \ell \\ p(k \mid \mu, s) & \text{if } \ell < k < u \\ 1 - F(u-1 \mid \mu, s) & \text{if } k = u \\ 0 & \text{otherwise} \end{cases}$$

where

$$p(k \mid \mu, s) = \frac{e^{1/s} - 1}{e^{1/s} + 1} \cdot e^{-|k-\mu|/s}$$

$$F(x \mid \mu, s) = \begin{cases} \frac{e^{1/s}}{e^{1/s}+1} \cdot e^{-(\mu - x)/s} & \text{if } x \le \mu \\ 1 - \frac{1}{e^{1/s}+1} \cdot e^{-(x-\mu)/s} & \text{if } x > \mu \end{cases}$$

If the support contains at least four points, then there exists a $k \in \{\ell, \ell+1, \ldots, \mu, \mu+1, \ldots, u\}$ such that:

$$\Pr[Z = k] \notin \left\{ \tfrac{t}{2^N} : t, N \in \mathbb{N} \right\}$$

That is, there is at least one output mass that is non-dyadic.

Proof. Let $q = e^{-1/s} \in (0, 1)$ and $c = \dfrac{1-q}{1+q}$. For every interior index $\ell < k < u$ we have

$$\Pr[Z = k] = c \cdot q^{|k-\mu|}$$

Case 1: q is dyadic. Write $q = m/2^N$ with $1 \le m < 2^N$ odd. Then

$$c = \frac{1-q}{1+q} = \frac{2^N - m}{2^N + m}$$

whose denominator $2^N + m$ is not a power of two, so c is non-dyadic. Because $\ell < \mu < u$, it follows that $\Pr[Z = \mu] = c$ is a non-dyadic mass.

Case 2: q is *not* dyadic. Write $q = r/d$ in lowest terms. Since q is not dyadic, the denominator d is not a power of two, so it contains at least one odd prime factor. Because the support has ≥ 4 points, either $\ell < \mu + 1 < u$ or $\ell < \mu - 1 < u$. Assume $\ell < \mu + 1 < u$ (the other side is symmetric). Then

$$\Pr[Z = \mu + 1] = c \cdot q = \frac{d-r}{d+r} \cdot \frac{r}{d}$$

Let j be the odd prime dividing d. Because $\gcd(r, d) = 1$, j does not divide r. Consequently j does not divide $(d - r)$ or $(d + r)$. Thus the denominator $d \cdot (d + r)$ contains at least one factor j, whereas the numerator $r \cdot (d - r)$ contains none. After cancelling the greatest common divisor, a power of j remains in the denominator, so the reduced fraction is not of the form $t/2^N$. Hence $\Pr[Z = \mu + 1]$ is non-dyadic.

□

Lemma A.3 (Finite-Coin Sampler for Censored Dyadic Symmetric Geometric)**.** Let integers $\ell < \mu < u$ and $p \in (0, 1)$ (dyadic rational) be given. Draw one unbiased coin for a sign $S \in \{-1, +1\}$ and draw $G \sim \text{Geom}(p)$. Define a Dyadic Symmetric Geometric random variable as:

$$Z = \begin{cases} \mu - G & \text{if } S = -1 \\ \mu + 1 + G & \text{if } S = +1 \end{cases}$$

Then a *Censored* Dyadic Symmetric Geometric random variable

$$Y = \max\{\ell, \min\{Z, u\}\}$$

is exactly sampleable with a finite (constant) number of unbiased coin flips under the $\text{RAM}_{\text{BDDNT}}$ model.

Proof. Flip one unbiased coin to choose a sign $S \in \{-1, +1\}$ and draw $G \sim \text{Geom}(p)\big|_0^m$, the geometric distribution with parameter $p = 2^{-k}$ clamped to $\{0, 1, \ldots, m\}$ where $m = \max\{\mu - \ell, u - \mu\}$. Each Bernoulli($p$) trial in the geometric sampler can be implemented using exactly k unbiased coins, so the clamped geometric draw uses at most $k \cdot (m + 1)$ coin flips. Together with the one coin for S, the entire sampling procedure uses at most $1 + k \cdot (m + 1)$ unbiased coins. Once S and G are drawn, define $Y = \mu - G$ if $S = -1$ and $Y = \mu + 1 + G$ if $S = +1$. The value Y is a sample from the clamped Dyadic Symmetric Geometric distribution centered at μ and clamped to $[\ell, u]$ as desired. Therefore, Y is sampleable with a finite (constant) number of unbiased coin flips under the $\text{RAM}_{\text{BDDNT}}$ model. □

Lemma A.4 (Censored Dyadic Symmetric Geometric Mechanism is DP)**.** Fix integers $\ell < \mu < u$ and let $\mu' = \mu + 1$. Let $p \in (0, 1)$ be any dyadic rational. Let $Z \sim \text{DTG}_{\mu,p}$ and $Z' \sim \text{DSG}_{\mu',p}$ be samples from the (unclamped) Dyadic Symmetric Geometric distribution centered at μ and μ', respectively. Let $Y = \max\{\ell, \min\{Z, u\}\}$ and $Y' = \max\{\ell, \min\{Z', u\}\}$. Then for every $y \in \mathbb{Z} \cap [\ell, u]$ we have

$$\frac{\Pr[Y = y]}{\Pr[Y' = y]} \le \frac{1}{1 - p} \quad \text{and} \quad \frac{\Pr[Y' = y]}{\Pr[Y = y]} \le \frac{1}{1 - p}$$

so the mechanism is ε-differentially private with

$$\varepsilon = \ln\left(\tfrac{1}{1-p}\right)$$

Proof. We analyze the *unclamped* Dyadic Symmetric Geometric random variables Z and Z'. Generate Z by flipping one unbiased coin for a sign $S \in \{-1, +1\}$ and drawing $G \sim \text{Geom}(p)$, setting $Z = \mu - G$ if $S = -1$ and $Z = \mu + 1 + G$ if $S = +1$. Generate Z' in the same way but with center $\mu' = \mu + 1$. For every $z \in \mathbb{Z}$ the shift of the center changes the probability mass by *at most* a single factor of $q = 1 - p$, so

$$\frac{\Pr[Z = z]}{\Pr[Z' = z]} \leq q^{-1} \quad \text{and} \quad \frac{\Pr[Z' = z]}{\Pr[Z = z]} \leq q^{-1}$$

Since the ratio for Z and Z' is bounded by $1/(1-p)$, the random variables Z and Z' are ε–differentially private with

$$\varepsilon = \ln\left(\tfrac{1}{1-p}\right)$$

The outputs $Y = \max\{\ell, \min\{Z, u\}\}$ and $Y' = \max\{\ell, \min\{Z', u\}\}$ are obtained from Z and Z' by deterministic clamping, which is a post-processing operation. Consequently the same bound holds for every $y \in \mathbb{Z} \cap [\ell, u]$ and the clamped mechanism is also ε–DP.

□

Theorem A.5 (Pure JOT-DP $\text{RAM}_{\text{BDDNT}}$ Programs in the Unbounded Setting)**.** For all $0 < \beta < 1$, $\varepsilon > 0$, $\varepsilon' > \varepsilon$, and ε-JOT-DP $\text{RAM}_{\text{BDDNT}}$ programs $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ in the *upper*-bounded setting, there exists a ε'-JOT-DP $\text{RAM}_{\text{BDDNT}}$ program $P' : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ for the unbounded setting such that

$$\left\| \texttt{out}(P(x, \texttt{env})) - \texttt{out}(P'(x, \texttt{env})) \right\|_{TV} < \beta$$

Furthermore, the mechanism $P'(x, \texttt{env})$ is simply an explicit algorithm that makes one oracle call to P on a truncation of x along with an additional computation that takes time $O(|x|)$ with probability at least $1 - \beta - e^{-\Omega(|x|)}$.

Proof. By replacing the call to `CensoredDiscreteLaplace` in Program 1 with the Censored Dyadic Symmetric Geometric Mechanism `CDSG`, we obtain an equivalent result for Theorem 3.4 in the $\text{RAM}_{\text{BDDNT}}$ model. This follows directly from Lemma A.3 and Lemma A.4 which give the needed ε_i-DP mechanism that executes in constant-time during each iteration of the program's loop. The rest of the proof follows identically to the proof of Theorem 3.4. In particular, during each iteration of the loop, when $m_i \geq (2/\varepsilon_i) \cdot \ln(1/\beta_i)$ we can round $p = 1 - e^{-\varepsilon_i}$ *down* to the nearest dyadic rational. Then when $m_i < |x|$, we have that for $\hat{n}_i = \texttt{CDSG}(\mu = m_i, p = 1 - e^{-\varepsilon_i}, \ell = 0, u = m_i)$:

$$\Pr\left[\hat{n}_i < \tfrac{m_i}{2}\right] = \frac{(1-p)^{\frac{m_i}{2}}}{2} \leq e^{-m_i \cdot \varepsilon_i / 2} \leq \beta_i$$

as desired. Furthermore, if we let iteration j be the first iteration when $m_j > 4 \cdot |x|$, then for $\hat{n}_j = \texttt{CDSG}(\mu = |x|, p = 1 - e^{-\varepsilon_j}, \ell = 0, u = m_j)$:

$$\Pr\left[\hat{n}_j > \tfrac{m_j}{2}\right] = \frac{(1-p)^{\frac{m_j}{2}-|x|}}{2} \leq e^{-\varepsilon_j \cdot |x|} \leq e^{-\Omega(|x|)}$$

Thus, with high probability the program halts on iteration j, and therefore the runtime analysis follows exactly to that of Theorem 3.4. □