# On the Feasibility of Forgetting in Data Streams

A. PAVAN ⓡ*, Iowa State University, USA
SOURAV CHAKRABORTY ⓡ, Indian Statistical Institute, India
N. V. VINODCHANDRAN ⓡ, University of Nebraska-Lincoln, USA
KULDEEP S. MEEL, University of Toronto, Canada

In today's digital age, it is becoming increasingly prevalent to retain digital footprints in the cloud indefinitely. Nonetheless, there is a valid argument that entities should have the authority to decide whether their personal data remains within a specific database or is expunged. Indeed, nations across the globe are increasingly enacting legislation to uphold the "Right To Be Forgotten" for individuals. Investigating computational challenges, including the formalization and implementation of this notion, is crucial due to its relevance in the domains of data privacy and management.

This work introduces a new streaming model: the 'Right to be Forgotten Data Streaming Model' (RFDS model). The main feature of this model is that any element in the stream has the right to have its history removed from the stream. Formally, the input is a stream of updates of the form $(a, \Delta)$ where $\Delta \in \{+, \perp\}$ and $a$ is an element from a universe $U$. When the update $\Delta = +$ occurs, the frequency of $a$, denoted as $f_a$, is incremented to $f_a + 1$. When the update $\Delta = \perp$, occurs, $f_a$ is set to 0. This feature, which represents the forget request, distinguishes the present model from existing data streaming models.

This work systematically investigates computational challenges that arise while incorporating the notion of the right to be forgotten. Our initial considerations reveal that even estimating $\mathsf{F}_1$ (sum of the frequencies of elements) of the stream is a non-trivial problem in this model. Based on the initial investigations, we focus on a modified model which we call $\alpha$-RFDS where we limit the number of forget operations to be at most $\alpha$ fraction. In this modified model, we focus on estimating $\mathsf{F}_0$ (number of distinct elements) and $\mathsf{F}_1$. We present algorithms and establish almost-matching lower bounds on the space complexity for these computational tasks.

CCS Concepts: • **Theory of computation → Streaming models**.

Additional Key Words and Phrases: data streams, right-to-forget, frequency moments

**ACM Reference Format:**
A. Pavan ⓡ, Sourav Chakraborty ⓡ, N. V. Vinodchandran ⓡ, and Kuldeep S. Meel. 2024. On the Feasibility of Forgetting in Data Streams . *Proc. ACM Manag. Data* 2, 2 (PODS), Article 102 (May 2024), 17 pages. https://doi.org/10.1145/3651603

*All authors contributed equally to this research. The symbol ⓡ denotes random author order. The publicly verifiable record of the randomization is available at https://www.aeaweb.org/journals/policies/random-author-order

Authors' addresses: A. Pavan ⓡ, pavan@cs.iastate.edu, Iowa State University, 226 Atanasoff Hall, Ames, Iowa, USA, 50010; Sourav Chakraborty ⓡ, chakraborty.sourav@gmail.com, Indian Statistical Institute, 203 Barrackpore Trunk Road, Kolkata, West Bengal, India, 700108; N. V. Vinodchandran ⓡ, vinod@unl.edu, University of Nebraska-Lincoln, 366 Avery Hall, Lincoln, Nebraska, USA, 68588; Kuldeep S. Meel, meel@cs.toronto.edu, University of Toronto, 40 St. George Street, Room 4283, Toronto, Ontario, Canada, M5S 2E4.


ACM 2836-6573/2024/5-ART102
https://doi.org/10.1145/3651603

## 1 INTRODUCTION

In the era of rapid and unprecedented digitization, data has emerged as a pivotal and invaluable asset. This significance is underscored by the widely acknowledged assertion in the digital domain that "Data Is the New Oil" [1]. Irrespective of the validity of equating data to oil, it is an incontrovertible truth that corporations globally are zealously accumulating and storing extensive volumes of personal data. The first two decades of the 21st century have seen a troubling trend where it seems that the digital activities and footprints of individuals and entities could be perpetually stored in the cloud.

In response to these growing concerns, governments and regulatory bodies worldwide have shifted their focus toward developing regulatory frameworks and legislations that empower individuals to regain control over the use of their personal data. A significant achievement in this effort is the European Union's enactment of the General Data Protection Regulation (GDPR), which grants individuals the 'Right to be Forgotten' [17]. Following this, other countries have considered implementing similar privacy-centric laws. While the concept of the Right to be Forgotten (RTBF) initially centered on search engines, recent advocacy by privacy activists has expanded its scope, calling for more comprehensive privacy laws. In this evolving context, it is crucial to explore the computational challenges associated with concepts like RTBF in various computational settings.

*This work initiates the exploration of the concept of forgetting within the data streaming model.* The streaming model is characterized by computations with constrained storage capacity, where data arrives as a continuous flow of elements. The objective is to develop algorithms with efficient update time complexity (i.e., the time required to process each stream element) and space complexity. The streaming model of computation has facilitated fundamental breakthroughs, leading to the deployment of systems in various domains such as data mining [6], network monitoring [13], security, and similar fields [15].

Our work's primary conceptual contribution is the introduction of a new streaming model: the *Right to be Forgotten Data Streaming Model* (RFDS model). This model empowers the end-user to execute a *forget* operation, enabling them to request, at any point, the omission of a specific data element, referred to as "$a$", from the stream. This requirement obligates the system to execute computations as if the element "$a$" had never been part of the stream. Users are permitted to make an unlimited number of such requests for any number of elements.

It is vital to underscore that the RFDS model diverges significantly from standard models such as the insertion and the turnstile model. For example, while the classic insertion model does not accommodate element deletion, the turnstile model does allow for the user-specified reduction in the frequency of an element. However, achieving a frequency reduction to zero necessitates that users track the element's frequency throughout, which contradicts the fundamental principle of streaming computation. This requirement would render it more practical for users to perform computations on their own.

We explore the classical problem of estimating frequency moments under the RFDS model. Our first set of results highlights a challenging aspect of frequency estimation within the RFDS model. Specifically, we demonstrate that in this model, any randomized algorithm estimating the $k^{th}$ frequency moment for any $k$ requires $\Omega(n)$ space, where $n$ is the size of the universe from which the stream elements originate.

Considering this somewhat pessimistic lower bound, one might naturally question the practical viability of the RFDS model. It is crucial to understand the basis of these robust lower bounds. In simpler terms, this negative outlook originates from scenarios where almost every user requests their data to be forgotten. This situation is analogous to banking systems facing significant challenges during a bank run, when a substantial proportion of customers try to withdraw their funds

simultaneously. Modern banking systems are designed to handle withdrawals up to a specific threshold of the bank's total deposits.

We introduce an $(\alpha, \mathsf{F}_k)$-RFDS model, ensuring that forget requests do not cause the corresponding frequency moment (of $\mathsf{F}_k$) to fall below a multiplicative factor of $(1-\alpha)$ of what the frequency moment would have been in the absence of forget requests. For example, in designing a data structure to monitor distinct elements (i.e., $\mathsf{F}_0$) for a website, our model guarantees that the proportion of customers making a forget request never exceeds an $\alpha$ fraction of the total number of distinct visitors.

In alignment with our desiderata, we demonstrate the feasibility of designing efficient algorithms within the $(\alpha, \mathsf{F}_k)$-RFDS model. We introduce efficient streaming algorithms for estimating the functions $\mathsf{F}_0$ and $\mathsf{F}_1$. For $\mathsf{F}_0$, we present an algorithm in the $(\alpha, \mathsf{F}_0)$-RFDS model with a space complexity of $O\left(\frac{1}{1-\alpha} \cdot \frac{\log n}{\varepsilon^2} \cdot \log \frac{1}{\delta}\right)$. Similarly, for $\mathsf{F}_1$, we propose an algorithm in the $(\alpha, \mathsf{F}_1)$-RFDS model, requiring $O\left(\frac{1}{1-\alpha} \cdot \frac{\log n}{\varepsilon^2} \cdot \log \frac{m}{\delta}\right)$ space complexity, where $m$ is the total number of updates in the stream. Further, we explore lower bounds, obtaining $\Omega\left(\frac{1}{\varepsilon^2(1-\alpha)} + \log n\right)$ for $\mathsf{F}_0$ estimation and $\Omega\left(\frac{1}{1-\alpha} \cdot \frac{1}{\varepsilon^2} + \log\log m\right)$ for $\mathsf{F}_1$ estimation in the $(\alpha, \mathsf{F}_0)$ and $(\alpha, \mathsf{F}_1)$ models respectively.

A primary motivation for our research was to evaluate the practicality of integrating the 'right to be forgotten' concept into streaming computations. This integration would enable regulatory bodies to mandate internet companies to adopt variants of the RFDS model in their systems. *Our key finding is that incorporating the 'right to be forgotten' in streaming systems is feasible, provided that forget requests are bounded.* Additionally, from a technical standpoint, the $(\alpha, \mathsf{F}_k)$-RFDS model presents intriguing technical challenges, particularly the notable gaps between the lower and upper bounds for both $\mathsf{F}_0$ and $\mathsf{F}_1$. Future research directions include investigating the possibility of achieving poly$(\log n)$-space complexity for $\mathsf{F}_2$ in the $(\alpha, \mathsf{F}_2)$-RFDS model, as well as higher moments.

We conclude this introduction with a brief discussion of earlier studied models that are directly related to RFDS model. The work of Hoffman, Muthukrishnan, and Raman [8] studied a model known as the *reset model*. Here each stream element is an update of the form $(a, v)$ implying that the frequency of $a$ in the stream is updated to $v$. It is easy to see that a forget operation can be simulated via a reset update: forget operation of an item $a$ can be replaced with an update $(a, 0)$. However, critically the reset model does not support insertion update (incrementing frequency by a positive value). Observe that RFDS model can simulate the reset model by first forgetting the item and then incrementing its frequency by insertions. Thus, the RFDS model generalizes the reset model. In addition, in [8], the authors restricted their focus to *monotone update model*, i.e. the frequency of an item never goes down. In particular, the forget operation is not supported in the monotone update model and their techniques and results are not applicable in the RFDS model. A recent work of Jayaram and Woodruff is related, but distinct, to the $\alpha$-RFDS model. They examined a variant of the turnstile model called the *bounded deletion* model [10], which limits the number of deletions in the turnstile model.

## 2 PRELIMINARIES

We denote the set $\{1, \ldots, n\}$ as $[n]$. For a string $a \in \{0,1\}^n$, the Hamming weight of $a$, represented as $|a|$, is the count of '1's in the string $a$. We define two well-established data stream models to provide a comprehensive framework for comparing and contrasting different stream models. Each stream element is considered a tuple $(a, \Delta)$, where $a$ belongs to the universe $U$ of size $n$, and $\Delta$ is an element of a constant sized set of actions. We assume the universe $U$ to be $[n]$ for simplicity.

For any element $a$ from the universe, its frequency in the stream is denoted by $f_a$, which is initially zero for all $a$. Upon arrival of a stream item $(a, \Delta)$, the frequency $f_a$ is updated based on $\Delta$. In the standard *insertion-only stream model*, the data stream $\mathcal{A} = \langle (a_1, \Delta_1), (a_2, \Delta_2), \ldots, (a_m, \Delta_m) \rangle$ comprises elements $a_i \in [n]$ and $\Delta_i \in \{+\}$. The frequency $f_a$ is incremented by 1 when $(a, \Delta)$ arrives. The *turnstile model*, another model relevant to our work, allows $\Delta \in \{+, -\}$ for each stream item $(a, \Delta)$. In this case, the frequency vector is updated as $f_a \leftarrow f_a - 1$ when $(a, -)$ arrives. Other related models include the strict-turnstile model and the cache register model. We refer to [5] for a detailed survey on various streaming models.

One of the most explored problems in data streaming models is frequency moment estimation. The $k$-th frequency moment $\mathsf{F}_k$ of the stream, defined as $\mathsf{F}_k = \sum_{a \in [n]} |f_a|^k$, is the $k$-th moment of the frequency vector $\langle f_1, \ldots, f_n \rangle$. This paper primarily focuses on $\mathsf{F}_0$ and $\mathsf{F}_1$, which count the number of items with non-zero frequency and the sum of all frequencies, respectively. Given $\varepsilon, \delta \in [0, 1]$, a streaming algorithm is an $(\varepsilon, \delta)$-estimator (or outputs an $(\varepsilon, \delta)$-estimate) of $\mathsf{F}_k$ of the stream if it produces a number Est at the stream's end such that, with at least $(1-\delta)$ probability, $\mathsf{Est} \in (1 \pm \varepsilon)\mathsf{F}_k$. Note that by employing standard techniques, a $(\varepsilon, 1/3)$-estimator of $\mathsf{F}_k$ can be amplified to a success probability of $1 - \delta$, with a multiplicative factor of $\log \frac{1}{\delta}$ in the space complexity, thus achieving an $(\varepsilon, \delta)$-estimator.

Pair-wise independent hash functions play a central role in the design of data stream algorithms.

*Definition 2.1.* A family of hash functions $\mathcal{H}(k)$ is 2-wise independent if, for all distinct $x_1, x_2 \in \{0,1\}^k$ and $\alpha_1, \alpha_2 \in \{0,1\}^k$, with $h \stackrel{R}{\leftarrow} \mathcal{H}(k)$, $\Pr[(h(x_1) = \alpha_1) \wedge (h(x_2) = \alpha_2)] = \frac{1}{2^{2k}}$

We use a standard explicit family of hash functions [4].

### 2.1 Communication Complexity

We employ communication complexity lower bounds to establish our streaming algorithms' space requirements. Typically, we use the one-way communication complexity model, where Alice and Bob jointly compute a function on their concatenated inputs, with the only communication being one-way from Alice to Bob. The following communication problem is of interest to us:

*Definition 2.2.* In the $\mathsf{GapANDindex}_{\varepsilon,r,d}$ problem, Alice is given $\mathbf{x} = x^1, \ldots, x^r \in \{0,1\}^d$, and Bob is given an index vector $\mathbf{i} = i_1, \ldots, i_r$ along with a bit vector $\mathbf{b} = b_1, \ldots, b_r$. They must compute (with at least 2/3 probability) the $\mathsf{GapANDindex}_{\varepsilon,r,d}(\mathbf{x}, \mathbf{i}, \mathbf{b})$, a partial function defined as:

$$\mathsf{GapANDindex}_{\varepsilon,r,d}(\mathbf{x}, \mathbf{i}, \mathbf{b}) = \begin{cases} 1 & \text{if } \sum_{j=1}^{r} (x^j_{i_j} \wedge b_j) \geq \frac{r}{4} + C\varepsilon r \\ 0 & \text{if } \sum_{j=1}^{r} (x^j_{i_j} \wedge b_j) \leq \frac{r}{4} - C\varepsilon r \end{cases}$$

This problem was introduced in [16], where they utilized the one-way communication complexity lower bound of the GapANDindex problem to derive a lower bound for the set intersection estimation problem. The authors established the following lower bound result, which we will use:

Theorem 2.3 ([16]). *The lower bound for the one-way communication complexity of the* $\mathsf{GapANDindex}_{\varepsilon,r,d}$ *problem, for the case* $r = \theta(1/\varepsilon^2)$*, is* $\Omega(dr)$*. This lower bound also holds with each* $x_i$ *having a Hamming weight of* $\Theta(d)$*.*

*Remark:* The work in [16] did not explicitly state that the lower bound holds when each $x_i$ has a Hamming weight of $\theta(d)$. However, their proof readily extends to this case, as the lower bound input instances for Alice and Bob are randomly constructed. With high probability, the Hamming weight of each $x_i$ is $\Theta(d)$, thus extending their proof to the case stated in the above theorem.

Another problem of interest in this paper is the well investigated Index problem:

*Definition 2.4.* In the Index problem, Alice has a set $A \subseteq [n]$, and Bob has $i \in [n]$. The task is to determine if $i \in A$ or not, with Alice sending a single message to Bob.

A straightforward algorithm is for Alice to send a characteristic string of $A$ to Bob using $n$ bits of communication. This is asymptotically the best strategy, even in a randomized setting.

Theorem 2.5 ([12]). *The randomized one-way communication complexity of* Index *is* $\Omega(n)$.

### 2.2 Concentration Inequalities

Let $X$ be a random variable. We denote $\mathbb{E}(X)$ and $\mathrm{Var}(X)$ as the expectation and variance of $X$, respectively. We will use the following three standard concentration inequalities to prove the correctness of our algorithms:

Theorem 2.6 (Markov's Inequality). *If $X$ is a non-negative random variable and $a > 0$, then* $\Pr[X \geq a] \leq \frac{\mathbb{E}(X)}{a}$.

Theorem 2.7 (Chebyshev's Inequality). *For any random variable $X$, and any positive real number $K$,* $\Pr[|X - \mathbb{E}(X)| \geq K] \leq \frac{\mathrm{Var}(X)}{K^2}$.

Theorem 2.8 (Chernoff Bound). *Suppose $v_1, ..., v_n$ are independent random variables taking values in $\{0, 1\}$. Let $V = \sum_{i=1}^{n} v_i$ and $\mu = \mathbb{E}[V]$. Then,*

$$\Pr(|V - \mu| \geq \delta\mu) \leq 2e^{-\frac{\delta^2\mu}{3}} \text{ for } 0 < \delta < 1$$

## 3 RFDS MODEL OF DATA STREAMING

We first formalize the *Right to Be Forgotten Data Streaming Model* (RFDS). In this model, the stream elements are members of a universe $U$ of size $n$. For any element $a \in U$, let $f_a$ denote its frequency, i.e., the number of times $a$ appears in the stream.

*Definition 3.1.* In the right to be forgotten data streaming model, input is a stream of updates of the form $(a, \Delta)$ where $\Delta \in \{+, \perp\}$. If $\Delta = +$, then the frequency of $a$, $f_a$, gets updated to $f_a + 1$. If $\Delta = \perp$, then $f_a$ is reset to 0.

*Comparison to Existing Models:* We now examine the relationship of the RFDS model with respect to existing models extensively studied in the literature: (1) the *insertion model*, (2) the *turnstile model*.

In the *insertion model*, the only permissible updates are of the form $(a, +)$. That is, each item of the stream is an element $a$ from the universe, and upon its arrival, the frequency $f_a$ is incremented to $f_a + 1$. Consequently, the Right to Be Forgotten model extends the insertion model.

In the *turnstile model*, the input stream comprises updates of the form $(a, \Delta)$ where $a$ is an element in the universe and $\Delta \in \{+, -\}$. If $\Delta = +$, $f_a$ is incremented to $f_a + 1$, and if $\Delta = -$, $f_a$ is decremented to $f_a - 1$. The key distinction between the RFDS model and the turnstile model lies in the nature of updates: while updates in the turnstile model are linear operations, in the RFDS model, the $\perp$ update is non-linear, eliminating every occurrence of the element $a$ and resetting its frequency to 0. This distinction is particularly evident when considering the computation of $\mathsf{F}_1$.

In insertion and turnstile streaming models, there is a very easy and space-efficient deterministic algorithm to exactly compute $\mathsf{F}_1$ that keeps a counter. However, we observe that in the RFDS model, surprisingly, $\mathsf{F}_1$ and $\mathsf{F}_0$ are computationally equivalent.

Proposition 3.2. *(1) $\mathsf{F}_0$ reduces to $\mathsf{F}_1$: There is a constant space streaming algorithm that takes a* RFDS *stream $\mathcal{D}$ and outputs another* RFDS *stream $\mathcal{D}'$ such that $\mathsf{F}_1(\mathcal{D}') = \mathsf{F}_0(\mathcal{D})$. (2) $\mathsf{F}_1$ reduces to $\mathsf{F}_0$: There is a $O(\log m)$-space streaming algorithm that takes a* RFDS *stream $\mathcal{D}$ as input and outputs a* RFDS *stream $\mathcal{D}'$ such that $\mathsf{F}_0(\mathcal{D}') = \mathsf{F}_1(\mathcal{D})$. Here $m$ is the number of updates of stream $\mathcal{D}$.*

Proof. For (1), let $\mathcal{D}$ be a RFDS stream. Construct a new stream $\mathcal{D}'$ as follows: For an update $(a, +)$ in $\mathcal{D}$, introduce updates $(a, \perp)$ and $(a, +)$ in $\mathcal{D}'$. For an update $(a, \perp)$ in $\mathcal{D}$, keep the same update in $\mathcal{D}'$. This algorithm operates in constant space. It is easy to see that $\mathsf{F}_1$ of $\mathcal{D}'$ is the same as $\mathsf{F}_0$ of $\mathcal{D}$. This is because all the occurrences of an item, other than its last occurrence, are deleted from $\mathcal{D}'$. For (2), construct a new stream as $\mathcal{D}'$ from $\mathcal{D}$ as follows: Maintain a counter $c$ that keeps track of the number of updates of the stream. When an update $(a, +)$ arrives, introduce update $(\langle a, c\rangle, +)$, where $c$ is the value of the counter. When $(a, \perp)$ arrives, introduce updates $(\langle a, 1\rangle, \perp) \ldots (\langle a, c-1\rangle, \perp)$ (this could blow the stream length by a quadratic factor)[1]. Again it is easy to see that $\mathsf{F}_0(\mathcal{D}')$ equals $\mathsf{F}_1(\mathcal{D})$. Since the conversion algorithm keeps track of the counter $c$, it uses $O(\log m)$ space. □

This suggests that RFDS model is different from existing models. Indeed, we establish that the randomized complexity of estimating frequency moments is asymptotically larger in the RFDS model than in the turnstile model. On the other hand, we also note that the deterministic complexity of estimating $\mathsf{F}_0$ (number of elements with non-zero frequency), is asymptotically smaller than the RFDS model than the turnstile model. Finally, note that we can augment the turnstile model with a forget operation to get a more general model, however, in this work we only consider augmenting the insertion model with a forget operation.

### 3.1 Initial Considerations

The model as defined, while very natural, has limitations. In particular, the fact that every element in the universe has the right to be forgotten gives rise to the situation where even deciding whether there is an element with non-zero frequency becomes difficult without essentially storing information about all possible elements. This leads to the result that estimating $\mathsf{F}_k$ for any $k$ requires $\Omega(n)$ space. This is in stark contrast to other models where sub-linear space algorithms are known for estimating frequency moments.

#### 3.1.1 *Randomized Complexity of Estimating $\mathsf{F}_k$ in the* RFDS *Model.*

Theorem 3.3. *Any randomized algorithm that decides whether the stream is empty (the frequency vector is the 0 vector) or not in the* RFDS *model requires $\Omega(n)$ space.*

We will use communication complexity as a tool to establish this lower bound. In particular, we will use the problem SetInclusion, which we define next.

*Definition 3.4 (*SetInclusion*).* Alice has a set $A$ and Bob has a set $B$ over a universe $[n]$. The goal is to decide whether $A \subseteq B$.

Fact 1. *The randomized one-way communication complexity of* SetInclusion *is $\Omega(n)$.*

Proof. We reduce Index to SetInclusion. An instance $(A, i)$ for Index will reduce to $(A, B)$ where $B = [n] \setminus \{i\}$. Now $i \notin A$ if and only if $A \subseteq B$. □

Proof. (*of Theorem 3.3*) Let $\mathcal{A}$ be an algorithm that decides whether an RFDS stream is empty or not that uses $s$ bits of space. We can use this algorithm to design a $s$-bit one-way communication protocol for SetInclusion as follows. Alice who has set $A$ will generate a stream of the form $\langle a, +\rangle$ for every $a \in A$ and simulate the algorithm $\mathcal{A}$ on this stream. Alice sends the state $S$ of the algorithm to Bob. Bob will generate the following stream: For every $b \in B$, generate a stream $\langle b, +\rangle$ followed by the stream $\langle b, \perp\rangle$. Now Bob will continue the simulation of algorithm $\mathcal{A}$ (starting from state $S$) on this stream. Bob declares that $A \subseteq B$ if an only if the output of algorithm $\mathcal{A}$ is 0. Observe that

[1]This reduction from $\mathsf{F}_1$ to $\mathsf{F}_0$ is pointed to us by an anonymous researcher.

the stream is empty if and only if $A \subseteq B$. Thus Bob can decide whether $A \subseteq B$ or not by $s$ bits of communication. Since randomized one-way communication complexity of SetInclusion is $\Omega(n)$, $s = \Omega(n)$. □

An immediate corollary to the above lower bound is that there are no space-efficient algorithms to estimate frequency moments in the RFDS model. This is because the frequency moments are non-zero if and only if the stream is non-empty. Thus if there is an algorithm with space $s$ for estimating the frequency moments within a multiplicative factor of $1 + \varepsilon$ for any $\varepsilon > 0$, then that algorithm can be used to decide the emptiness of the stream.

Corollary 3.5. *For any $k$, any randomized algorithm that estimates $\mathsf{F}_k$ in the* RFDS *requires $\Omega(n)$ space.*

In particular, randomized space complexity of $\mathsf{F}_0$ estimation in the RFDS model is $\Omega(n)$. However, it is known that in turnstile model $\mathsf{F}_0$ can be estimated using $\text{poly}(\log n)$ space (when the approximation parameter $\varepsilon$ and the confidence parameter $\delta$ is constant) [3, 7, 9]. Thus for randomized $F_0$ estimation, there is an exponential gap between the turnstile and RFDS models.

### 3.2 Deterministic Complexity of Estimating $\mathsf{F}_0$: Turnstile vs RFDS

We investigate the deterministic space complexity of estimating $\mathsf{F}_0$ in RFDS model. Somewhat surprisingly we show that in the deterministic realm, the space complexity in RFDS model is slightly smaller than the space complexity in the turnstile model. In the RFDS model, $\mathsf{F}_0$ can be computed exactly using $O(n)$ space. We next prove that the deterministic complexity of estimating $\mathsf{F}_0$ in the turnstile model is $\Theta(n \log m)$. This result is perhaps known in the community. However, we have not encountered a proof. Hence we present it here.

Theorem 3.6. *Deterministic complexity of computing $\mathsf{F}_0$ of a stream of length $m$ in* RFDS *model is $\Theta(n)$ where $n$ is the size of the universe. On the other hand, any deterministic algorithm that estimates $\mathsf{F}_0$ in the turnstile model (support of the frequency vector) within a relative error of $1/4$ requires $\Omega(n \log m)$ space.*

Proof. In the RFDS model, we can store an $n$-bit vector that keeps track of whether the frequency of each item is 0 or not. Start with all 0 vector and then if an update + comes flip the corresponding bit to 1. If a $\perp$ comes, set it back to 0. The lowerbound of $\Omega(n)$ follows from the result in the earlier section where it is shown that randomized space complexity (even for approximating $\mathsf{F}_0$) is $\Omega(n)$.

Now we show that any deterministic algorithm for estimating $\mathsf{F}_0$ within a relative error of $1/4$ in the turnstile model requires $\Omega(n \log m)$ space. Consider the following promise problem $(\Pi_y, \Pi_n)$. $\Pi_y = \{\mathcal{D} \mid \mathsf{F}_0(\mathcal{D}) \leq n/2\}$ and $\Pi_n = \{\mathcal{D} \mid \mathsf{F}_0(\mathcal{D}) = n\}$. Any deterministic algorithm that estimates $\mathsf{F}_0$ within a relative error of $1/4$ can be easily modified to accept all strings from $\Pi_y$ and reject all strings from $\Pi_n$. Let $A$ be a streaming algorithm that solves this promise problem.

The technique is to model the algorithm $A$ that uses $s$ space as a deterministic finite state automaton $\mathcal{F}_A$ with $2^s$ states and alphabet $\Sigma = [n] \times \{-1, +1\}$. In this automaton, for two states $s_1$ and $s_2$, there is a transition from $s_1$ to $s_2$ if the algorithm $A$ with memory state $s_1$ on update $(a, b)$ changes the memory state to $s_2$. Consider those states where $A$ accepts as accepting states.

We will use the Myhil-Nerode theorem to prove the lower bound. For two strings, $x, y$ over $\Sigma$, we say $x \equiv y$ if for all strings $z$ over $\Sigma$, $\mathcal{F}_A$ accepts $xz$ if and only if it accepts $yz$. Then by Myhil-Nerode theorem, the number of states of $\mathcal{F}_A$ is at least the number of equivalence classes of this equivalence relation.

We will construct a set of strings where each string belongs to different equivalence classes. For this, consider the following fact.

Fact 2. *Let $n$ be an integer and $p > n$ be a prime number. For any large enough $n$, there is set of $S$ consisting of of $n$-tuples of the form $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle$ where each $\ell_i \in [p]$ with the following properties.*

*(1) $|S| \geq p^{n/2}$.*

*(2) Any distinct pair of tuples $t, r \in S$ differ at least in $n/2$ places.*

Proof of the above fact can be established via error-correcting codes, specifically Reed-Solomon codes. Let $q(x)$ be the polynomial of degree $n/2$ over the field $GF(p)$. For each such polynomial, associate a tuple $t$ obtained by evaluating the polynomial $q$ on the first $n$ elements of the field. Two different polynomials give two different tuples. Since any two polynomials of degree $n/2$ evaluate to the same value at fewer than $n/2$ places, it follows that tuples corresponding to two distinct polynomials differ in at least $n/2$ places. Since the number of such polynomials is $\geq p^{n/2}$, the above statement follows.

For every tuple $t = \langle \ell_1, \ell_2, \cdots, \ell_n \rangle \in S$, we associate a $x_t$ a string over the alphabet of $\mathcal{F}_A$ which is $\Sigma = [n] \times \{-1, +1\}$. The string $x_t$ has $\ell_i$ many symbols of the form $\langle i, + \rangle$ $\ell_i$ times, for $1 \leq i \leq n$. We will argue that for two distinct tuples $t = \langle \ell_1 \ldots \ell_n \rangle$ and $r = \langle \ell'_1 \ldots \ell'_n \rangle$, the strings $x_t$ and $x_r$ belong to two different equivalence classes of $\mathcal{F}_A$. Since $t$ and $r$ are distinct tuples from $S$, they differ in at least $n/2$ indices. Let $I$ be the set of such indices, note $|I| \geq n/2$. Consider the string $z$ defined as follows: For every $j \in I$, $z$ contains the symbol $\langle j, - \rangle$ $l_j$ times. Now consider the strings $x_t z$ and $x_r z$. Notice that $\mathsf{F}_0(x_t z) \leq n/2$ and $\mathsf{F}_0(x_r z) = n$. Thus $x_t$ and $x_r$ belong to two different equivalence classes. Since the size of $S$ is $p^{n/2}$, it follows that the number of equivalence classes is $p^{n/2}$. Thus the number of states of $\mathcal{F}_A$ is at least $p^{n/2}$. This implies that the $s \geq \frac{n}{2} \log p$. Note that the size of the stream $m \leq 2pn$. Since $p > n$, this gives a $\Omega(n \log m)$ lower bound on the space complexity in the turnstile model. □

### 3.3 A Refinement: The Bounded Forget Model

Building upon the findings earlier in this section, it becomes evident that in the RFDS model, no substantial statistics of the stream can be computed in sub-linear space. The primary reason for this limitation is that the model accommodates requests equating to 'forgetting all the data.' However, as highlighted in the introduction, realistic scenarios typically involve only a fraction of the data being subject to forget requests. This observation leads us to propose a refined version of the model, termed the $\alpha$-RFDS model. This new model is conceptualized within the context of computing a generic statistical function G over the frequency vector of the data stream.

Let $\mathcal{D}$ be a stream in the RFDS model. Let $\mathcal{D}'$ be the stream obtained by removing all the forget updates. The *forget-factor* of $\mathcal{D}$ with respect to G is defined as $\mathrm{G}(\mathcal{D})/\mathrm{G}(\mathcal{D}')$.

*Definition 3.7 ($(\alpha, \mathrm{G})$-RFDS model).* For a parameter $\alpha : 0 \leq \alpha < 1$ and a statistics G, a stream $\mathcal{D}$ is a $(\alpha, \mathrm{G})$-RFDS stream if the forget factor of $\mathcal{D}$ with respect to G is $\geq 1 - \alpha$.

In this work, much of our focus is on estimating the $\mathsf{F}_0$ and $\mathsf{F}_1$ in the stream. Thus, we will focus on the $(\alpha, \mathsf{F}_0)$-RFDS and the $(\alpha, \mathsf{F}_1)$-RFDS models.

To reiterate, in the $(\alpha, \mathsf{F}_0)$-RFDS model, at the end of the stream, it is guaranteed that at most $\alpha$-fraction of the distinct elements seen in the stream will be forgotten. On the other hand, in the $(\alpha, \mathsf{F}_1)$-RFDS model, at the end of the stream, it is guaranteed that at most $\alpha$-fraction of the elements seen in the stream will be forgotten. For the sake of brevity when the statistic under consideration (that is $\mathsf{F}_0$ and $\mathsf{F}_1$) is clear from the context we will call the model the $\alpha$-RFDS model. So $\mathsf{F}_0$ estimation in the $\alpha$-RFDS model would actually mean $\mathsf{F}_0$ estimation in the $(\alpha, \mathsf{F}_0)$-RFDS model. Simlarly $\mathsf{F}_1$ estimation in the $\alpha$-RFDS model means $\mathsf{F}_1$ estimation in the $(\alpha, \mathsf{F}_1)$-RFDS model.

**Remark.** As mentioned, in the RFDS model, $F_0$ and $F_1$ are reducible to each other. However, these reductions do not preserve the forget factor, and we do not know of reductions between these two

problems that preserve the forget factor. Thus we need to establish upper and lower bounds for $F_0$ and $F_1$ separately in the $\alpha$-RFDS model.

## 4 $\mathsf{F}_0$ ESTIMATION IN THE $\alpha$-RFDS MODEL

In this section, we study the complexity of the classic problem of estimating the number of distinct elements in the $\alpha$-RFDS model. Our main contribution is an upper bound on the space complexity and a matching lower bound with respect to $\alpha$ and $\varepsilon$.

### 4.1 Algorithm for $\mathsf{F}_0$ estimation in the $\alpha$-RFDS Model

Theorem 4.1. *There is an algorithm* $\mathsf{F}_0\mathsf{Estimate}$ *that given inputs* $\varepsilon \in (0,1)$ *and a stream* $\mathcal{D}$ *in the* $(\alpha, \mathsf{F}_0)$-RFDS *model, outputs a value* Est *such that*

$$\Pr\left[(1-\varepsilon)\mathsf{F}_0(\mathcal{D}) \leq \mathsf{Est} \leq (1+\varepsilon)\mathsf{F}_0(\mathcal{D})\right] \geq \frac{2}{3}.$$

*The algorithm* $\mathsf{F}_0\mathsf{Estimate}$ *uses at most* $O\left(\frac{1}{1-\alpha} \cdot \frac{\log n}{\varepsilon^2}\right)$ *space.*

The algorithm and its proof closely follow the BJKST algorithm for $\mathsf{F}_0$ estimation in the insertion-only model [3, 7]. The main idea of the BJKST algorithm is to ensure that every distinct element in the stream is selected into a bucket with a certain probability $p$. The value of $p$ is adaptively changed so that (1) it is small enough to limit the number of elements selected into the bucket to ensure that the space complexity is bounded, and (2) it is large enough so that the number of elements selected is at least $1/\varepsilon^2$ to ensure a $(1 \pm \varepsilon)$ multiplicative estimate of $\mathsf{F}_0$. To satisfy condition (1), the value of $p$ is adaptively decreased as soon as the bucket size crosses a certain threshold. The threshold in BJKST is set so that at any point in the stream, the value of $p$ is around $1/(K\varepsilon^2)$, where $K$ is the $\mathsf{F}_0$ of the stream so far, and this satisfies the requirement (2).

In the case of the $\alpha$-RFDS model, the $\mathsf{F}_0$ of the stream can suddenly drop by a factor of $\alpha$. This may occur when all the forget updates follow all the insertion updates. For such a stream, after the insertion updates, the bucket size is $O(\frac{1}{\varepsilon^2})$. When a forget update arrives, the forgotten element (if exists) should be removed from the bucket. However, this causes the expected number of selected elements to fall below the $1/\varepsilon^2$ by a factor of $(1-\alpha)$. This violates requirement (2), leading to an error in the final estimate. To compensate for this sudden drop in $\mathsf{F}_0$, we have to ensure that the value of $p$ always has a slack by a factor of $\frac{1}{1-\alpha}$. We accomplish this by increasing the threshold by a factor of $\frac{1}{1-\alpha}$.

Proof. (Of Theorem 4.1)

We follow the proof outline given in [5]. In the following for a binary string $x$, $\mathsf{Zero}(x)$ is the number of leading zeros of $x$.

Let $X_{a,r}$ be a RV defined as follows: If $\mathsf{Zeros}(h(a)) \geq r$, then $X_{a,r} = 1$, else $X_{a,r} = 0$. Let $Y_r = \sum_{a, f_a > 0} X_{a,r}$. Let $z^*$ be the sampling level (value of $z$) when the algorithm terminates. Note that the output of the algorithm is $Y_{z^*} 2^{z^*}$.

Let $d$ be the value of $\mathsf{F}_0$ at the end of the stream and let $d'$ be the $\mathsf{F}_0$ value of the stream if we discard all $(\cdot, \perp)$ from the stream. Note that $d \geq (1-\alpha)d'$. Let $s$ be a sampling level such that the following holds

$$\frac{12}{\varepsilon^2} \leq \frac{(1-\alpha)d'}{2^s} \leq \frac{24}{\varepsilon^2}$$

The goal is to bound the probability that $|Y_{z^*} 2^{z^*} - d| \geq \varepsilon d$. Note that $E[Y_r] = \frac{d}{2^r}$ for all $r$. Thus

$$\Pr[|Y_{z^*} 2^{z^*} - d| \geq \varepsilon d] = \Pr[|Y_{z^*} - \frac{d}{2^{z^*}}| \geq \frac{\varepsilon d}{2^{z^*}}]$$

**Algorithm 1** $\mathsf{F}_0\mathsf{Estimate}(\mathcal{D}, \varepsilon, \alpha)$

Pick $h : [n] \to [n]$ from a pair-wise independent hash family
$\mathcal{B} \leftarrow \emptyset$, $\mathsf{Thresh} \leftarrow \frac{1}{(1-\alpha)} \cdot \frac{c}{\varepsilon^2}$, $z \leftarrow 0$
**while** not End-of-Stream **do**
    Upon update $(a, \Delta)$
    **if** $\Delta = \perp$ **then** $\mathcal{B} \leftarrow \mathcal{B} \setminus \{a\}$
    **else if** $\mathsf{Zero}(h(a)) \geq z$ **then** $\mathcal{B} \leftarrow \mathcal{B} \cup \{a\}$
    **if** $|\mathcal{B}| > \mathsf{Thresh}$ **then**
        $z \leftarrow z + 1$
        Remove all $a$s from $\mathcal{B}$ for which $\mathsf{Zero}(h(a)) > z$
**Output** $|\mathcal{B}|2^z$

$$\Pr\left[|Y_{z^*} - \frac{d}{2^{z^*}}| \geq \frac{\varepsilon d}{2^{z^*}}\right] = \sum_{r=0}^{r=\log n} \Pr\left[\left(|Y_r = \frac{d}{2^r}| \geq \frac{\varepsilon d}{2^r}\right) \cap (z^* = r)\right]$$

$$\leq \sum_{r=1}^{s-1} \Pr\left[|Y_r - \frac{d}{2^r}| \geq \frac{\varepsilon d}{2^r}\right] + \sum_{r=s}^{\log n} \Pr[z^* = r]$$

$$= \sum_{r=1}^{s-1} \Pr\left[|Y_r - E[Y_r]| \geq \frac{\varepsilon d}{2^r}\right] + \Pr\left[Y_{s-1} \geq \frac{c}{(1-\alpha)\varepsilon^2}\right]$$

We will bound both parts separately. Bounding the first sum:

$$\sum_{r=1}^{s-1} \Pr\left[|Y_r - E[Y_r]| \geq \frac{\varepsilon d}{2^r}\right] \leq \sum_{r=1}^{s-1} \mathrm{Var}(Y_r) \cdot \frac{2^{2r}}{\varepsilon^2 d^2}$$

$$\leq \sum_{r=1}^{s-1} \frac{d}{2^r} \cdot \frac{2^{2r}}{\varepsilon^2 d^2} \leq \frac{2^s}{\varepsilon^2 d} \leq \frac{1}{\varepsilon^2} \cdot \frac{2^s}{d'(1-\alpha)} \leq \frac{1}{12}$$

To bound the second quantity, we will rely on Markov Inequality,

$$\Pr[Y_{s-1} \geq \frac{c}{(1-\alpha)\varepsilon^2}] \leq E[Y_{s-1}] \cdot \frac{(1-\alpha)\varepsilon^2}{c}$$

$$= \frac{d}{2^{s-1}} \cdot \frac{(1-\alpha)\varepsilon^2}{c} \leq \frac{d'}{2^{s-1}} \cdot \frac{(1-\alpha)\varepsilon^2}{c}$$

$$\leq \frac{48}{c} \leq \frac{1}{5} \text{ (for } c = 240)$$

□

The above algorithm and analysis gives a constant probability of success. This can be amplified to $1 - \delta$ probability with a multiplicative factor of $\log \frac{1}{\delta}$ using standard techniques. We remark that we could use a double hashing technique as is done by the BJKST algorithm ([3]) to further reduce the space complexity. We formally state the space complexity below and present a proof sketch.

Theorem 4.2. *The algorithm* $\mathsf{F}_0\mathsf{Estimate}$ *can be modified so that its space complexity is*

$$O\left(\log n + \frac{1}{(1-\alpha)\varepsilon^2}\left(\log\log n + \log\frac{1}{\varepsilon} + \log\frac{1}{1-\alpha}\right)\right).$$

Proof Sketch. The main space usage of algorithm $\mathsf{F}_0$Estimate is the storage of bucket $\mathcal{B}$ which keeps $O(\frac{1}{(1-\alpha)} \cdot \frac{1}{\varepsilon^2})$ elements of the universe and each element needs $O(\log n)$ space. The idea to reduce the space is to store the hash values of elements (in the bucket), as opposed to the elements themselves. The main observation is that the expected number of elements, during the execution of the algorithm, that can get into the bucket is $O(\mathsf{Thresh} \cdot \frac{\log n}{1-\alpha})$. Thus we pick another pairwise independent hash function $g : [n] \to [10 \cdot (\mathsf{Thresh} \cdot \frac{\log n}{1-\alpha})^2]$. When an update $\langle a, + \rangle$ arrives, compute $h(a)$ and $g(a)$ and store $\langle g(a), \mathsf{Zeros}(h(a)) \rangle$ in the bucket $\mathcal{B}$. When $\langle a, \perp \rangle$ arrives, remove $\langle g(a), \cdot \rangle$ from the bucket. The correctness proof proceeds exactly as before, except that we have to account for the collisions due to the hash function $g$. Since the range of $g$ is large enough, the collision probability can be bounded by $1/10$. Thus the probability that we output a good approximation will be $2/3 - 1/10$. The total space is determined by: $O(\log n)$ to store the hash functions $h$ and $g$, $O(\log \log n + \log \frac{1}{\varepsilon} + \log \frac{1}{1-\alpha})$ to store each element (of the form $\langle g(a), \mathsf{Zeros}(h(a)) \rangle$) of the bucket.

### 4.2 Lower Bound of $\mathsf{F}_0$ Estimation in the $\alpha$-RFDS Model

In this section, we establish that the upper bound of $O(\frac{1}{1-\alpha} \frac{\log n}{\varepsilon^2})$ is optimal with respect to the parameters $\alpha$ and $\varepsilon$.

Theorem 4.3. *Any streaming algorithm that computes a $(\varepsilon, 1/3)$-approximation of $\mathsf{F}_0$ in the $(\alpha, \mathsf{F}_0)$-*RFDS *model uses a space of $\Omega(\frac{1}{1-\alpha} \cdot \frac{1}{\varepsilon^2} + \log n)$.*

The $\Omega(\log n)$ bound comes from the same known lower bound for $\mathsf{F}_0$ in the insertion model. Hence we will focus on $\Omega(\frac{1}{1-\alpha} \cdot \frac{1}{\varepsilon^2})$ bound. We will use communication complexity bound to establish this. One of the most well-studied problems in communication complexity is the set disjointness problem, where Alice and Bob are given two sets (from the universe $\Omega$) and they have to check if their sets are disjoint. One of the celebrated works in communication complexity states that the communication complexity of the set disjointness problem is $\Theta(|\Omega|)$ [12]. A harder problem is the set intersection problem, where Alice is given a set $A$ and Bob is given a set $B$ and they have to compute the size of $|A \cap B|$. A related problem to set intersection estimation problem is about estimating the size of the intersection of the two sets within a factor of $(1 \pm \varepsilon)$. It is easy to see that these variants - the set intersection and set intersections estimation problems - are harder than the set disjointness problem and hence has communication complexity $\Theta(|\Omega|)$. But if one is guaranteed an additional condition that the size of $|A \cap B|$ is known to be bigger than some quantity (say $(1-\alpha)|\Omega|$), then the problem becomes much easier. In this paper, we use this problem as a stepping stone to prove the lower bound on the $\alpha$-RFDS model.

*Definition 4.4 (*Large-Set-Intersection-Estimation). Alice has a set $A \subseteq \Omega$ and Bob has a set $B \subseteq \Omega$ and it is promised that $|A| \geq \eta|\Omega|$ and $|A \cap B| \geq (1-\alpha)|A| \geq (1-\alpha)\eta|\Omega|$. The task is to output an estimate Est such that $(1-\varepsilon)|A \cap B| \leq \mathsf{Est} \leq (1+\varepsilon)|A \cap B|$, with probability $2/3$.

Before we obtain bounds on the communication complexity of the Large-Set-Intersection-Estimation Problem we show that a lower bound on the communication on Large-Set-Intersection-Estimation translates to a lower bound on the space complexity of the $\alpha$-RFDS model.

Lemma 4.5. *The one-way communication complexity of the* Large-Set-Intersection-Estimation *is a lower bound on the space complexity of the $\alpha$-*RFDS *model.*

Proof. Let $A$ and $B$ be an instance of Large-Set-Intersection-Estimation. That is $A$ and $B$ are two subsets of $\Omega$ such that $|A| \geq \eta|\Omega|$ and $\eta$ is some constant. Consider the following stream: (1) The first $|A|$ elements of the stream are of the form $(a, +)$ for all $a \in A$, (2) The next $|\overline{B}|$ elements of

the stream are of the form $(b, +)$ for all $b \notin B$, and (3) The next $|\overline{B}|$ elements of the stream are of the form $(b, \perp)$ for all $b \notin B$.

In other words, all the elements of $A$ have their frequency first increased by 1, elements of $\overline{B}$ have their frequency first increased by 1 and then all the elements of $\overline{B}$ is forgotten. Note that in the end the $\mathsf{F}_0$ of the stream is $|A \cap B|$. While the total number of distinct elements seen in the stream is $|A \cup \overline{B}|$ which is at least $\eta|\Omega|$ and $|A \cap B| \geq (1-\alpha)\eta|\Omega|$.

So if we assume that Alice sees the first $|A|$ number of items in the stream and Bob sees rest - then the problem of $\varepsilon$-estimation of $\mathsf{F}_0$ of the stream reduces to estimating (within $(1 \pm \varepsilon)$ the size of $|A \cap B|$ when Alice and Bob are given sets $A$ and $B$ respectively and it is promised that $|A \cap B| \geq (1-\alpha)\eta|\Omega|$.

Thus as long as $\eta$ is some constant a lower bound of $\frac{1}{(1-\alpha)\eta\varepsilon^2}$ on the communication complexity of Large-Set-Intersection-Estimation gives an asymptotically same lower bound to the space complexity for $\varepsilon$-estimation of $\mathsf{F}_0$ in the $\alpha$-RFDS model. □

One can get an upper bound of $\frac{1}{\varepsilon^2(1-\alpha)}$ on the communication complexity for Large-Set-Intersection-Estimation. But we are interested in lower bound of $\frac{1}{\varepsilon^2(1-\alpha)}$ on the communication complexity. We state the required theorem below. The proof is in the Appendix.

Theorem 4.6. *Alice has set $A$ and Bob has set $B$ with $A, B \subset \Omega$ with the following guarantees:*

- $|A| \geq |\Omega|/3$
- $|A \cap B| \geq (1-\alpha)|A \cup \overline{B}|$

*The communication of complexity of computing the $\varepsilon$ approximation of $|A \cap B|$ is $\Omega\left(\frac{1}{\varepsilon^2(1-\alpha)}\right)$.*

Proof. We will prove the theorem by reducing an instance of $\mathsf{GapANDindex}_{\varepsilon,r,d}$ (where $r = \Theta(1/\varepsilon^2)$ and $d = 1/(1-\alpha)$) to the Large-Set-Intersection-Estimation problem.

Assume that Alice is given $\mathbf{x} = x^1, \ldots, x^r \in \{0,1\}^d$ and Bob is given index vector $\mathbf{i} = i_1, \ldots, i_r$ along with a bit vector $\mathbf{b} = b_1, \ldots, b_r$ and they have to compute $\mathsf{GapANDindex}_{\varepsilon,r,d}(\mathbf{x}, \mathbf{i}, \mathbf{b})$. Consider the following reduction.

- $\Omega := \{d^i + j \ : \ \text{where } i \in [r] \text{ and } j \in [d]\}$.
- Alice and Bob constructs sets $A, B \subseteq \Omega$ respectively.
- Alice constructs $A$ as $A := \left\{d^i + j \mid x^i_j = 1\right\}$
- Bob constructs $B$ as $B := \left\{d^j + i_j \mid b_j = 1\right\}$

Note the following properties of $A$ and $B$,

- $|\Omega| = dr$
- $|A|$ is the number of 1s in $\mathbf{x}$ which is $\Theta(dr)$
- $|A \cap B| = \sum_{j=1}^{r}(x^j_{i_j} \wedge b_j)$. So we are guaranteed that $|A \cap B| = \Theta(r) = \Theta(\frac{|A|}{d})$ which is $\Theta(|A \cup \overline{B}|/d)$ as $|A| \geq \Omega/3$.
- So if $\mathsf{GapANDindex}_{\varepsilon,r,d}(\mathbf{x}, \mathbf{i}, \mathbf{b}) = 1$ then $|A \cap B| \geq \frac{r}{4} + C\varepsilon r$, and
- if $\mathsf{GapANDindex}_{\varepsilon,r,d}(\mathbf{x}, \mathbf{i}, \mathbf{b}) = 0$ then $|A \cap B| \leq \frac{r}{4} - C\varepsilon r$
- And if one can estimate the $|A \cap B|$ within an additive error of $C\varepsilon r$ then one can compute $\mathsf{GapANDindex}_{\varepsilon,r,d}(\mathbf{x}, \mathbf{i}, \mathbf{b})$

Hence by Theorem 2.3 we have that the communication of complexity of computing the $\varepsilon$ approximation of $|A \cap B|$ is $\Omega(\frac{d}{\varepsilon^2})$. Thus we have our theorem.

□

## 5 $\mathsf{F}_1$ ESTIMATION IN THE $\alpha$-RFDS MODEL

In this section, we delve into the complexity of estimating the number of elements, specifically $\mathsf{F}_1$, in the $\alpha$-RFDS model. Contrasting with standard models such as the insertion-only and turnstile models, where computing $\mathsf{F}_1$ of a stream is straightforward using only $\log m$ space (where $m$ is the number of elements in the stream), the task is significantly more complex in the RFDS model. We demonstrate in Section 5.2 that estimating $\mathsf{F}_1$ in the RFDS model is as challenging as estimating $\mathsf{F}_0$ in both the RFDS and $\alpha$-RFDS models. This establishes a lower bound on the space complexity for computing an $(\varepsilon, 1/3)$-estimate of $\mathsf{F}_0$ in the $\alpha$-RFDS model. On the other hand, we present an upper bound on the space complexity, along with a matching lower bound, in relation to $\alpha$ and $\varepsilon$.

### 5.1 Algorithm

Theorem 5.1. *There is an algorithm* $\mathsf{F}_1\mathsf{Estimate}$ *that given inputs* $\varepsilon \in (0,1)$ *and a stream* $\mathcal{D}$ *in the* $(\alpha, \mathsf{F}_1)$-RFDS *model and an upper bound,* $m$*, on the total number of updates in the stream* $\mathcal{D}$*, outputs a value* Est *such that*

$$\Pr\left[(1-\varepsilon)\mathsf{F}_1(\mathcal{D}) \le \mathsf{Est} \le (1+\varepsilon)\mathsf{F}_1(\mathcal{D})\right] \ge \frac{2}{3}.$$

*The algorithm* $\mathsf{F}_1\mathsf{Estimate}$ *uses at most* $O\left(\frac{1}{1-\alpha} \cdot \frac{\log n}{\varepsilon^2} \cdot \log m\right)$ *space.*

**Algorithm 2** $\mathsf{F}_1\mathsf{Estimate}(\mathcal{D}, \varepsilon, \alpha)$

1: $\mathcal{B} \leftarrow \phi$, $\mathsf{Thresh} \leftarrow \lceil \frac{12}{\varepsilon^2(1-\alpha)} \log(24m) \rceil$, $p \leftarrow 1$
2: **while** not End-of-Stream **do**
3: Upon seeing $(a, \Delta)$
4: **if** $\Delta = \perp$ **then** Remove all copies of $a$ from $\mathcal{B}$
5: **if** $\Delta = +$ **then**
6: With probability $p$, add a copy of $a$ to $\mathcal{B}$
7: **while** $|\mathcal{B}| = \mathsf{Thresh}$ **do**
8: $p \leftarrow \frac{p}{2}$
9: **for** $a \in B$ **do**
10: Remove $a$ from $\mathcal{B}$ with probability 1/2
11: **Output** $|\mathcal{B}|/p$

Proof. The algorithm and its proof closely follow the sampling-based algorithm of [14] for $\mathsf{F}_0$ estimation in the insertion model.

Firstly, observe that the algorithm $\mathsf{F}_1\mathsf{Estimate}$ stores at most $\mathsf{Thresh}$ number of items (counting different number of copies of the same item) in $\mathcal{B}$. Since one needs $\log n$ number of bits to store the information of one item in the stream, so the space complexity of the algorithm is at most $\mathsf{Thresh} \cdot \log n$, as claimed in the theorem statement.

Before we continue to prove the correctness of the algorithm $\mathsf{F}_1\mathsf{Estimate}$, one may observe that there is a non-zero probability of the algorithm not halting, the **while** loop in lines 7 to 10 may not terminate. We will prove later that this can be appropriately fixed. We will observe that the stopping probability follows the geometric distribution with a expectation less than 1 and also low variance. So if one makes the algorithm halt, abort, and output $\perp$ after one round of the **while** loop, we can prove that the probability of the **while** loop(s) in lines 7 to 10 not terminating after 1 round. And this would imply that the probability that the algorithm $\mathsf{F}_1\mathsf{Estimate}$ (as stated) and the modified algorithm differ is very low.

So, let us now prove the correctness of the algorithm $\mathsf{F}_1$Estimate. Consider the event:

Error : 'The algorithm $\mathsf{F}_1$Estimate does not return a value in the range $[(1-\varepsilon)\mathsf{F}_1, (1+\varepsilon)\mathsf{F}_1]$'

We will show $\Pr[\mathsf{Error}] \leq \frac{1}{6}$.

Intuitively the algorithm at any point stores a sample of items that are in the stream. The idea is that, at any point in time each copy of each element of the stream (after removal of those that have been forgotten) is independently sampled with probability $p$ and stored in $\mathcal{B}$. So at the end the expected size of $\mathcal{B}$ is $\mathsf{F}_1 \cdot p$. So one expects that $|\mathcal{B}|/p$ would be a good estimate of the $\mathsf{F}_1$. The last statement is true and can be formalized appropriately if the value of $p$ is not less than $\log(24m)/\varepsilon^2\mathsf{F}_1$. The value of Thresh is chosen such that this bound on $p$ can be ensured.

Observe that the algorithm gives a joint distribution over the state of $\mathcal{B}$ and the value of $p$. To prove that every element of the stream (after removal of those that has been forgotten) is independently sampled with probability $p$ and stored in $\mathcal{B}$ we have to formalize it carefully.

To obtain the desired bound on $\Pr[\mathsf{Error}]$ let us denote by $Y_j^i$ the set $\mathcal{B}$ after $i$ elements have been seen in the stream and when $p = 2^{-j}$. Note that there are two moves that happen:

- Go from $Y_j^i$ to $Y_j^{i+1}$ (Line 4 and 6)
- Go from $Y_j^i$ to $Y_{j+1}^i$ (Line 7 to 10)

When the algorithm moves from $Y_j^i$ to $Y_j^{i+1}$, it can happen either in Line 4 or in Line 6. In Line 4 we throw away all traces of the elements that are forgotten. And in Line 6 we pick the $(i+1)$th element with probability $p$. Thus, for any fixed $j$, every entry in the stream (that are not forgotten) so far (upto the $i$th element of the stream) is in $Y_j^i$ independently with probability $1/2^j$.

Now, let Bad be the event "The value of $p$ at line 11 in Algorithm 2 is less than $\frac{\mathsf{Thresh}(1-\alpha)}{4\mathsf{F}_1}$." Note that the value of $p$ goes down by $1/2$ every time the size of $\mathcal{B}$ is equal to Thresh. Consider the first time the value of $p$ goes down below $\frac{\mathcal{T}\langle\nabla\rceil\int\langle(1-\alpha)}{4\mathsf{F}_1}$. So at that time the expected number of elements in $\mathcal{B}$ must is at most $\frac{\mathsf{Thresh}(1-\alpha)}{4\mathsf{F}_1}$ times the number of elements that has ever come in the stream, which is at most $\mathsf{F}_1/(1-\alpha)$. In other words, when the first time the value of $p$ goes down below $\frac{\mathsf{Thresh}(1-\alpha)}{4\mathsf{F}_1}$ the expected number of elements in $\mathcal{B}$ is at most $\frac{\mathsf{Thresh}(1-\alpha)}{4\mathsf{F}_1} \cdot \frac{\mathsf{F}_1}{1-\alpha}$ which is Thresh/4. Using Chernoff Bound (Theorem 2.8) and Union Bound we can bound the probability of event Bad to be at most $1/12$.

Also by Chernoff bound (Theorem 2.8) we observe that if the value of $p$ is at least $\frac{\varepsilon^{-2}\log 12}{\mathsf{F}_1}$ then the probability of event $\mathsf{Error} \cap \overline{\mathsf{Bad}}$ is at most $1/12$. Thus $\Pr[\mathsf{Error}] \leq 1/6$. □

*Ensuring that the algorithm always halts.* As noted earlier while the algorithm outputs an $(\varepsilon, 1/3)$-estimate of $\mathsf{F}_1$, the runtime of the algorithm is not bounded in the worst case. But that can be fixed easily by ensuring that the **while** loop (lines 7 to 10) exists after one iteration. The modified algorithm is presented in Algorithm 3.

Note that the Algorithm 2 and 3 on differ in case when the size of $\mathcal{B}$ does not go below Thresh after every item of $\mathcal{B}$ is thrown away independently with probability $1/2$.

Consider the event, Fail : 'The algorithm $\mathsf{F}_1$-Estimate outputs $\perp$.'

We will prove that $\Pr[\mathsf{Fail}] \leq \frac{1}{24}$. Also note that what we proved in the correctness of Algorithm 2 is the same as the correctness of Algorithm 3 when Fail does not happen.

Thus showing that $\Pr[\mathsf{Fail}]$ is at most $\frac{1}{24}$ suffices, since this coupled with the fact that $\Pr[\mathsf{Error}] \leq \frac{1}{6}$, implies the correctness of Algorithm 3. So we conclude the proof by showing $\Pr[\mathsf{Fail}] \leq \frac{1}{24}$.

Let $\mathsf{Fail}_j$ denote the event that Algorithm 3 returns $\perp$ when $i = j$. Formally, $\mathsf{Fail}_j$: '$|\mathcal{B}| = \mathsf{Thresh}$ and none of the elements of $\mathcal{B}$ are thrown away at line 10 for $i = j$'. The probability that $\mathsf{Fail}_j$ happens is $\left(\frac{1}{2}\right)^{\mathsf{Thresh}}$. Therefore,$\Pr[\mathsf{Fail}] \leq \sum_{j=1}^{m} \Pr[\mathsf{Fail}_j] \leq m \cdot \left(\frac{1}{2}\right)^{\mathsf{Thresh}} \leq \frac{1}{24}$.

**Algorithm 3** Modified-$\mathsf{F}_1$Estimate($\mathcal{D}, \varepsilon, \alpha$)

$\mathcal{B} \leftarrow \phi$, Thresh $\leftarrow \lceil \frac{12}{\varepsilon^2(1-\alpha)} \log(24m) \rceil$, $p \leftarrow 1$
**while** not End-of-Stream **do**
  Upon seeing $(a, \Delta)$
  **if** $\Delta = \perp$ **then** Remove all copies of $a$ from $\mathcal{B}$
  **if** $\Delta = +$ **then**
    With probability $p$, add a copy of $a$ to $\mathcal{B}$
  **if** $|\mathcal{B}|$ = Thresh **then**
    $p \leftarrow \frac{p}{2}$
    **for** $a \in B$ **do**
      Remove $a$ from $\mathcal{B}$ with probability 1/2
    **if** $|\mathcal{B}|$ = Thresh **then**
      Return $\perp$
**Output** $|\mathcal{B}|/p$

### 5.2 A Lower Bound for $\mathsf{F}_1$ in the $\alpha$-RFDS Model

We show that the lower bound of estimating $\mathsf{F}_0$ (without $\Omega(\log n)$) in the $\alpha$-RFDS model carries over to the estimation of $\mathsf{F}_1$ also. Proof of the following theorem appears in the appendix.

Theorem 5.2. *Any streaming algorithm that computes a $(\varepsilon, 2/3)$-approximation of $\mathsf{F}_1$ in the $(\alpha, \mathsf{F}_1)$-RFDS model uses a space of $\Omega(\frac{1}{1-\alpha} \cdot \frac{1}{\varepsilon^2} + \log\log m)$ .*

Proof. The main observation is that the streams for which we proved space lower bounds for $\mathsf{F}_0$ have all the elements distinct and hence has the same $\mathsf{F}_1$. Moreover, the forget-factor with respect to $\mathsf{F}_0$ is only a constant factor different from the forget-factor with respect to $\mathsf{F}_1$. Recall that the streams for which we established the lower bound for $\mathsf{F}_0$ is the following. For two subsets $A$ and $B$ of $\Omega$ such that $|A| \geq \eta|\Omega|$ for some constant $\eta$, the stream $\mathcal{D}$ is the following:

- The first $|A|$ elements of the stream are of the form $(a, +)$ for all $a \in A$
- The next $|\bar{B}|$ elements of the stream are of the form $(b, +)$ for all $b \notin B$.
- The next $|\bar{B}|$ elements of the stream are of the form $(b, \perp)$ for all $b \notin B$.

Thus the stream consists of elements of $A \cap B$ exactly once. Hence $\mathsf{F}_1(D) = \mathsf{F}_0(D)$. Moreover, we are also promised that forget factor of $\mathcal{D}$ is at least $(1-\alpha)$ (that is, $|A \cap B| \geq (1-\alpha)|A \cup \overline{B}|$). Now let us compare the forget factors with respect to both $\mathsf{F}_1$ and $\mathsf{F}_0$. Let $\mathcal{D}'$ be the stream in which all the forget updates (updates of the form $(b, \perp)$) are removed. Note that $\mathsf{F}_1(\mathcal{D}') \leq 2\mathsf{F}_0(\mathcal{D}')$. Thus $\mathsf{F}_1(D) = \mathsf{F}_0(D) \geq (1-\alpha)\mathsf{F}_0(D) \geq \frac{(1-\alpha)}{2}\mathsf{F}_1(D')$. Let $1-\kappa = (1-\alpha)/2$. Thus if there is an algorithm for estimating $\mathsf{F}_1$ with space complexity $o(\frac{1}{1-\kappa} \cdot \frac{1}{\varepsilon^2})$, the same algorithm also gives an estimate of $\mathsf{F}_0$ with space complexity $o(\frac{1}{1-\kappa} \cdot \frac{1}{\varepsilon^2}) = o(\frac{1}{1-\alpha} \cdot \frac{1}{\varepsilon^2})$ which is a contradiction to the lower bound for $\mathsf{F}_0$. □

## 6 CONCLUSION

In this paper, our objective was to formulate a model that embodies the 'Right to be Forgotten' concept within streaming computation. This task required a careful balance between adhering to the concept's essence and ensuring its applicability in algorithmic design. To evaluate the effectiveness of our formalization, we focused on the fundamental problem of frequency estimation, particularly $\mathsf{F}_0$ and $\mathsf{F}_1$. Initially, we demonstrated the impracticality of creating efficient data structures that can handle an unlimited number of "forget" requests. Inspired by the existing data streaming literature

and the design principles of contemporary banking systems, we proposed the $\alpha$-RFDS model. This model limits forget requests to a maximum fraction of $\alpha$. The $\alpha$-RFDS model introduces unique technical challenges. For example, unlike the straightforward computation of $\mathsf{F}_1$ in standard models like insertion and turnstile, computing $\mathsf{F}_1$ in the $\alpha$-RFDS model is non-trivial. Additionally, our lower and upper bounds for $\mathsf{F}_0$ prompt further inquiries for frequency moments. We highlight some of these research questions as follows:

**Optimal Bound for $\mathsf{F}_0$:** We established a lower bound of $\Omega(\log n + \frac{1}{\varepsilon^2(1-\alpha)})$ and an upper bound of $\tilde{O}(\log n + \frac{\log\log n}{\varepsilon^2(1-\alpha)})$. A pertinent question is how to narrow this gap. Similar efforts in the insertion model took considerable time and culminated in a significant result by Kane, Nelson, and Woodruff [11].

**Optimal Bound for $\mathsf{F}_1$:** The lower bound for $\mathsf{F}_1$ stands at $\Omega(\frac{1}{1-\alpha}\cdot\frac{1}{\varepsilon^2} + \log\log m)$, while the upper bound is $O\left(\frac{1}{1-\alpha}\cdot\frac{\log n}{\varepsilon^2}\cdot\log m\right)$. A significant line of future research would be to bridge this gap between the lower and upper bounds.

**Poly-Logarithmic Algorithms for $\mathsf{F}_2$:** The sampling-based method proposed by Alon, Matias, and Szegedy[2] for $\mathsf{F}_k$ (where $k > 1$) might be adapted to the $\alpha$-RFDS model, potentially leading to algorithms with dependence on $n^{1-1/k}$. A captivating question arises: is it possible to develop a $\mathrm{poly}(\log n)$ space algorithm for $\mathsf{F}_2$ in the $\alpha$-RFDS model, akin to other models?

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for constructive comments that improved the presentation and pointing to some related prior work. Pavan's work is partially supported by NSF grants 2130536 and 2342245. Vinodchandran's work is partially supported by the NSF grants 2130608 and 2342244. Kuldeep's work was partly supported in part by the National Research Foundation Singapore under its NRF Fellowship programme [NRF-NRFFAI1-2019-0004] and Campus for Research Excellence and Technological Enterprise (CREATE) programme, as well as the Ministry of Education Singapore Tier 1 and 2 grants R-252-000-B59-114 and MOE-T2EP20121-0011.

## REFERENCES

[1] 2017. The world's most valuable resource is no longer oil, but data. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

[2] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 20–29.

[3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2022. Counting Distinct Elements in a Data Stream. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2483)*, José D. P. Rolim and Salil P. VadMuth (Eds.). 1–10.

[4] J Lawrence Carter and Mark N Wegman. 1977. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*. ACM, 106–112.

[5] Amit Chakrabarti. 2023. Data Stream Algorithms, Lecture Notes. https://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf

[6] Graham Cormode, Minos Garofalakis, Peter J Haas, Chris Jermaine, et al. 2011. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases* 4, 1–3 (2011), 1–294.

[7] Phillip B. Gibbons and Srikanta Tirthapura. 2001. Estimating simple functions on the union of data streams. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, Arnold L. Rosenberg (Ed.). ACM, 281–291.

[8] Michael Hoffmann, S Muthukrishnan, and Rajeev Raman. 2007. Streaming algorithms for data in motion. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 294–304.

[9] Piotr Indyk. 2006. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53, 3 (2006), 307–323.

[10] Rajesh Jayaram and David P. Woodruff. 2018. Data Streams with bounded deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, Jan Van den Bussche and Marcelo Arenas (Eds.). ACM, 341–354.
[11] Daniel M Kane, Jelani Nelson, and David P Woodruff. 2010. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 41–52.
[12] Ilan Kremer, Noam Nisan, and Dana Ron. 1995. On randomized one-round communication complexity. In *STOC*, Frank Thomson Leighton and Allan Borodin (Eds.). ACM, 596–605. https://doi.org/10.1145/225058.225277
[13] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. 2006. Data streaming algorithms for estimating entropy of network traffic. *ACM SIGMETRICS Performance Evaluation Review* 34, 1 (2006), 145–156.
[14] Kuldeep S Meel, N. V. Vinodchandran, and Sourav Chakraborty. 2021. Estimating the size of union of sets in streaming models. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 126–137.
[15] Shanmugavelayutham Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science* 1, 2 (2005), 117–236.
[16] Rasmus Pagh, Morten Stöckel, and David P. Woodruff. 2014. Is min-wise hashing optimal for summarizing set intersection?. In *PODS*, Richard Hull and Martin Grohe (Eds.). 109–120. https://doi.org/10.1145/2594538.2594554
[17] The Europian Parliament and the Council of the Europian Union. 2022. Regulation (EU) 2016/679 of the European parliament and of the council. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 Accessed on 19 Sep 2023.

Received December 2023; revised February 2024; accepted March 2024