Electronic Colloquium on Computational Complexity, Report No. 60 (2026)

White-Box Adversarial Streaming Lower Bounds beyond Two-Party Communication

Klim Efremenko* Ben-Gurion University

Gillat Kol† Princeton University

Raghuvansh R. Saxena‡ Tata Institute of Fundamental Research

Zhijun Zhang§ INSAIT, Sofia University "St. Kliment Ohridski"

Abstract

Streaming algorithms in adversarial settings have attracted considerable attention recently. We show that, in the *white-box adversarial streaming model* [ABJ+22], the fundamental problem of estimating the F_p *moment* to within *any constant factor* requires $\Omega(n)$ memory. In this model, the internal state of the (randomized) streaming algorithm is visible to an adversary, who can exploit this information when constructing subsequent stream updates. As a corollary, we also obtain a white-box lower bound for the well-studied problem of *estimating the maximum matching size* in graphs.

[ABJ+22] proved that two-party white-box communication protocols can be derandomized. This allows them to prove *deterministic* communication lower bounds and automatically derive white-box (communication and streaming) lower bounds. However, such two-party lower bounds can only rule out approximation of the F_p moment within a specific constant factor. Ruling out approximation within *any constant factor* typically requires proving a lower bound for a *multi-party* communication problem.

We show that white-box communication protocols involving *any number of parties* can be derandomized, provided they compute a *total function*. However, this derandomization fails entirely when extended to partial functions and, consequently, to approximation problems. We are therefore compelled to prove our moment estimation lower bound for the white-box model directly. Our proof introduces a novel hybrid technique that, instead of taking hybrids over input distributions, constructs hybrids over white-box adversaries.

*klimefrem@gmail.com. Supported by the Israel Science Foundation (ISF) through grant No. 1456/18 and European Research Council Grant number: 949707.

†gillat.kol@gmail.com. Supported by a National Science Foundation CAREER award CCF-1750443 and by a BSF grant No. 2018325.

‡raghuvansh.saxena@gmail.com. Supported by the Department of Atomic Energy, Government of India, under project no. RTI4001.

§zhijun.zhang@insait.ai. This research was partially done while at Princeton University and was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

ISSN 1433-8092

Contents

1 Introduction

The classical streaming model. In the classical streaming model, an algorithm receives a sequence of elements $u_1, \ldots, u_m$, one at a time, and needs to output a value that depends on the stream while using limited memory. The output should be correct (with high probability) for any possible input stream. For example, the elements might represent entries in a database, and the algorithm may be tasked with estimating the number of distinct elements. The estimation must be correct for any content of the database and any order in which the entries appear.

Although this model requires the algorithm to work for worst-case input streams, potentially chosen by an adversary, it assumes that the entire stream is fixed in advance and that the adversary is *oblivious* to the algorithm's internal state, including its memory contents and random coins. For deterministic algorithms, this restriction is inconsequential, since the algorithm's behavior is entirely determined by the input stream. However, most useful streaming algorithms are randomized, and their internal state depends on both the input stream and the random coins used during execution.

Therefore, the classical assumption that the adversary cannot observe the algorithm's internal state effectively requires the input stream to be independent of the algorithm's randomness. This assumption has been challenged by many recent works that considered stronger adversaries capable of generating the input stream *adaptively*, based on (partial) knowledge of the algorithm's internal state.

White-box streaming model. The *white-box streaming model*, introduced by [ABJ+22], considers adversaries that see the full internal state of the streaming algorithm throughout its execution. Specifically, a white-box adversary can observe all previously used random coins (and, consequently, also the algorithm's memory contents) and may use this information to decide on the next stream element.

Formally, let π be a streaming algorithm. At time $t \in [m]$, a *white-box adversary* generates the element u_t based on the prior stream elements $u_1, \ldots, u_{t-1}$ and the previously sampled random strings $r_1, \ldots, r_{t-1}$. The algorithm then draws fresh randomness r_t and updates its internal memory using u_t, r_t, and its current memory contents[1]. For $\delta \in [0, 1]$, we say that π *solves* or *computes* a relation f with probability δ, if, for every white-box adversary, the algorithm's final output is correct with probability at least δ, over both the adversary's choices and the algorithm's randomness[2].

The white-box streaming model has analogues in several other areas. In the study of *dynamic data structures*, recent work has considered adversaries that generate updates adap-

[1] In the original formulation by [ABJ+22], the stream may consist of both insertions and deletions and the algorithm is required to output the current value of the function or relation after each update. In this work, we focus on the insertion-only setting where the output is only produced at the end. This only makes our lower bounds stronger.

[2] If the success probability is omitted, we take $\delta = 2/3$.

tively after observing the full contents of the data structure. Similarly, in *machine learning*, many recent efforts aim to design algorithms that are robust to adversarial white-box attacks, where attackers have full access to the model's parameters, architecture, or training model. In the context of *persistent data structures*, which are commonly used to enable efficient access to historical versions of shared data in collaborative environments, users often interact with exposed versioning mechanisms, allowing updates to depend on prior versions. For a more comprehensive survey of white-box models across different domains, see [ABJ+22, FW23].

The study of the white-box streaming model is motivated by several factors. First, algorithms may not have access to a securely private source of randomness, or they may run on remote machines where memory privacy cannot be guaranteed. Second, as with other public-coin algorithms (*e.g.*, public-coin interactive proofs), white-box algorithms support public verifiability. For example, if the algorithm publishes its internal state and uses a trusted, publicly available source of randomness, users contributing elements to the data stream can verify that the computations involving their data were performed correctly. A third motivation, highlighted by [ABJ+22, FW23], is that the white-box model captures scenarios where a centralized server distributes an initial state (*e.g.*, random bits or parameters) to a network of remote users. These users may then generate their data based on this shared state and send it back to the server, for example, for aggregate statistical analysis.

Moment estimation. Frequency moment estimation is among the most extensively studied problems in classical streaming algorithms, dating back to the seminal work of [AMS99]. Given a stream $u \in [n]^m$, let f_i denote the number of occurrences (frequency) of element $i \in [n]$ in the stream. The p-th frequency moment is defined as $F_p(u) = \sum_{i \in [n]} f_i^p$.

Estimating F_p has a wide range of applications, including efficient approximation of statistical properties, data summarization, and anomaly detection in high-throughput data environments such as those arising in computer network monitoring and security, sensor networks, social platforms, financial systems, and real-time analytics.

Elegant, randomized, streaming algorithms for α-approximation of F_p are known for any constant $\alpha > 1$ and any p. When $p \leq 2$, only $\tilde{O}(1)$ bits of memory are needed [AMS99, Ind06]. For $p > 2$, the space complexity of estimating F_p is $\tilde{\Theta}(n^{1-2/p})$ [BJKS04, IW05]. In contrast, for any $p \neq 1$, it is known that *deterministic* algorithms require $\Omega(n)$ memory to achieve any constant-factor approximation [CK16].

The gap between the memory requirements of randomized and deterministic algorithms, combined with the fact that white-box algorithms lie somewhere in between, raises the question: *Can F_p still be approximated using sublinear memory in the white-box streaming model for constant α?*

The many randomized streaming algorithms for moment estimation rely on a diverse set of algorithmic techniques or "tricks", each leveraging randomness in a different way. Designing white-box adversaries to break these algorithms may seem to require custom strategies.

For example, F_0, the number of distinct elements, can be estimated using hash functions,

e.g., [FM85, AMS99]. Intuitively, the algorithm hashes each stream element and maintains the minimum hash value. The smaller this hash value, the greater the estimated number of distinct elements. Observe that a white-box adversary can fail this algorithm by repeatedly feeding it with the same u that has the minimum hash value under the algorithm's chosen hash function.

Another technique for estimating F_0 relies on *sub-sampling*, *e.g.*, [CVM22]. Intuitively, the algorithm maintains a dynamic set, initially empty. Upon receiving an element u_i, if u_i is already in the set, it is removed; otherwise, it is (re-)inserted with a fixed probability p. A larger final set size corresponds to a higher estimate of the number of distinct elements. To break this algorithm, a white-box adversary can repeatedly insert a single value u_1 into the stream until the algorithm removes it from the set, then do the same with another element u_2, and so on, ensuring the set remains empty.

For general F_p estimation with $p \geq 1$, [AMS99] introduced an algorithm that selects a random element from the stream and estimates F_p by counting the number of times this element appears in the remainder of the stream. The higher this count, the higher the estimated F_p. To break the algorithm, a white-box adversary can repeatedly insert a single element u until it is selected. Then, the adversary only inserts a different element $u' \neq u$, ensuring a zero count for the selected element u.

1.1 Our Results

Lower bound for white-box moment estimation. Are any of the clever tricks used by randomized moment estimation streaming algorithms resilient to all white-box adversaries? Theorem 1.1 answers this question in the negative, proving that no white-box streaming algorithm using $o(n)$ memory can achieve any constant-factor approximation for any F_p.

Theorem 1.1. *For $p \neq 1$, $n \in \mathbb{N}$, and $\alpha = \alpha(n) \geq 1$, any white-box streaming algorithm that α-approximates F_p on $[n]$ with constant success probability greater than $1/2$, requires $\Omega_p\left(\alpha^{-\frac{\max(p,1)}{|p-1|}} \cdot n - O_p(\log n)\right)$ memory.*[3]

We note that for every p and α, the lower bound in Theorem 1.1 matches the known deterministic lower bound for F_p estimation [CK16]. For $p \in [0, 1)$ and any α, it also matches the deterministic upper bound [CK16][4].

We also note that a lower bound for white-box F_p estimation with weaker guarantees was previously shown by [ABJ+22]. Specifically, they ruled out white-box algorithms using $o(n)$ space for *some* constant α_p, whereas our result rules out $o(n)$-space algorithms for *any* constant approximation factor α, including arbitrarily large values. Moreover, our lower bound degrades gracefully for super-constant values of α.

[3]The notation Ω_p and O_p suppress constants in p, the asymptotic is in α and n. The $O_p(\log n)$ term is relevant only for large super-constant α.

[4][CK16] shows that for every p, there exists a deterministic streaming algorithm that α-approximates F_p over streams on $[n]$, using $\tilde{O}\left(\alpha^{-1/|p-1|} \cdot n\right)$ memory (for streams of length $\text{poly}(n)$).

Lower bound for approximating the maximum matching size. As a corollary of Theorem 1.1, we obtain a white-box space lower bound for α-approximating the *Maximum Matching Size* (MMS). The bound follows by a direct, simple reduction from our F_0 lower bound.

Theorem 1.2. *For every $n \in \mathbb{N}$ and every $\alpha = \alpha(n) \geq 1$, any white-box streaming algorithm that α-approximates MMS on undirected graphs on n vertices with constant success probability greater than $1/2$ requires $\Omega\left(\frac{n}{\alpha} - O(\log n)\right)$ memory.*

Quantitatively, Theorem 1.2 matches the best known deterministic lower bound [CK16]. A deterministic 2-approximation for MMS is straightforward with $O(n)$ space: maintain a maximal matching and output its size. In contrast, it is still open whether randomized streaming algorithms can achieve any constant-factor approximation to MMS using sublinear space. Currently, even n^{ε}-space algorithms cannot be ruled out, for any $\varepsilon > 0$.

For super-constant approximation factors, efficient randomized algorithms are known. In particular, when $\alpha = O(\sqrt{n})$, a folklore approach uses only $\operatorname{poly}\log(n)$ space (see, *e.g.*, [AKL17, KKS14]). In this regime, Theorem 1.2 gives an $\Omega(\sqrt{n})$ lower bound for white-box streaming algorithms, thus *separating randomized and white-box space complexity for approximating MMS*.

Derandomization of white-box algorithms for total functions. To prove their inapproximability result for F_p, [ABJ+22] gave a general theorem showing that for any total function and any boolean (possibly partial) function $f(x_1, x_2)$, every randomized white-box two-party communication protocol[5] can be converted to a *deterministic* protocol with the same communication cost.

Informally, white-box two-party communication protocols correspond to streaming algorithms that are resilient against an adversary who adapts *once* at a fixed point during the stream. To complete their proof, they gave an $\Omega(n)$ deterministic communication lower bound for the two-party decision version of the F_p problem.

While many classical streaming lower bounds are derived via reductions from two-party communication problems, such reductions often fall short for approximation tasks. For example, in the case of distinct elements, Alice and Bob can individually count the number of distinct elements in their respective inputs and simply add the counts. This yields a 2-approximation with little communication. Similarly, for sufficiently large α, a lower bound for α-approximation of F_p cannot be derived from a two-party communication lower bound.

[ABJ+22] also shows that their argument cannot be extended to general k-party communication problems. In contrast, our next theorem proved in Section 5 shows that for

[5]The white-box communication complexity of $f(x_1, \ldots, x_k)$ is the maximum communication per party of the best randomized, k-party protocol for f that is resilient to any white-box adversary. Such an adversary generates the input x_i for party i after observing the random strings of the previous $i-1$ parties. Upon receiving x_i, the message from party $i-1$, and a fresh random string, party i sends a message to party $i+1$. See Section 3.2.

protocols computing *total* functions with any number of parties, such a derandomization *is* possible. Informally, this means that any deterministic streaming memory lower bound for a total function also holds in the white-box streaming model.

Theorem 1.3. *For $k \geq 1$ and a* total *function $f(x_1, \ldots, x_k)$ (not necessarily boolean), if there is a white-box communication protocol that solves f with probability better than $1/2$, then there is also a one-way deterministic communication protocol that solves f with the same communication.*

Separating white-box from deterministic streaming. Theorems 1.1 and 1.3, along with the derandomization result of [ABJ+22], indicate that for many streaming problems, the white-box and deterministic memory requirements are effectively the same. Additionally, the best known separations between the two models are quantitatively small (at least for streams of polynomial length). For example, [ABJ+22] show that the deterministic space complexity of the approximate counting problem is $\Theta(\log n)$, whereas the Morris counter algorithm, which uses only $\Theta(\log \log n)$ space, can be implemented in the white-box setting[6].

Can white-box streaming algorithms use substantially less memory than deterministic ones? Theorem 1.3 rules out such separations for total functions. However, our next theorem proved in Section 6 shows that large separations *do* exist for (boolean) partial functions[7].

Theorem 1.4. *There is a boolean* partial *function f for which there exists a white-box streaming algorithm that computes it using $\tilde{O}(1)$ memory, while any deterministic algorithm that computes it uses $\tilde{\Omega}(n)$ memory.*

At a high level, the partial function f used in the proof of Theorem 1.4 forces the adversary to "commit" to part of the stream by imposing a promise that couples the prefix and the suffix of the stream. Consequently, even if the adversary can identify future stream updates that would cause the algorithm to fail, it cannot introduce them without violating the promise.

1.2 Additional Related Work

Several alternatives to the classical streaming model have recently received significant attention. One such extensively studied model, that is closely related to the white-box streaming model, is the *adversarially robust streaming model* [BY20, BEJW+22]. In this model, the streaming algorithm must respond to a query after each update (insertion or deletion), and the adversary may select the next update based on the algorithm's response. For example,

[6] We note that in models with computationally bounded adversaries, significantly larger separations have been shown under cryptographic assumptions [ABJ+22, FW23]. However, in this work, we focus on the information-theoretic notion of white-box adversaries.

[7] Such separations are trivially possible for sampling problems. For instance, outputting a random bit or a random element from the stream can be done with $O(1)$ space by a white-box algorithm, but makes little sense for deterministic streaming algorithms.

to α-approximate F_0, the algorithm must, after each update, output (with high probability) an α-approximation of the number of distinct elements seen so far. The adversary can then use this information to adaptively determine the next update.

Clearly, any algorithm that works in the white-box streaming model also works in the adversarially robust model. The robust model can be viewed as a "black-box" adversarial streaming model, since the adversary does not have access to the algorithm's internal state, only to the outputs it reveals through queries.

In contrast to our Theorem 1.1, which shows that for any constant $\alpha \geq 1$, $\Omega(n)$ memory is required to α-approximate F_p in the white-box insertion-only model, [BEJW+22] design algorithms that use sublinear space for the same task in the adversarially robust insertion-only model. The memory requirements of their algorithms match those of classical randomized algorithms for F_p estimation (up to logarithmic factors). These algorithms work by changing their responses to queries only a small number of times, thereby revealing only a limited amount of their randomness. However, they fail entirely if the adversary is given access to the full internal state of the algorithm.

A major open problem posed by [BEJW+22] is to determine the space complexity of adversarially robust F_p-estimation in the turnstile streaming model. A recent result by [GLW+25] makes substantial progress by ruling out linear sketches. Note that since the white-box lower bound in Theorem 1.1 is proved for insertion-only streams, it also applies to the white-box turnstile model.

The white-box streaming model, introduced by [ABJ+22], captures a richer class of adversarial scenarios. As discussed above, [ABJ+22] prove lower bounds for this model that are derived from *deterministic* two-party communication lower bounds. On the algorithmic side, the significant power granted to adversaries in the white-box model has led most research to focus on settings where adversaries are computationally bounded, often under cryptographic assumptions. Surprisingly, despite the fact that streaming algorithms in the white-box model lack even a secret key, [ABJ+22, FW23] were able to leverage cryptographic tools to design space-efficient algorithms for a range of classical problems, including L_1-heavy hitters, turnstile L_0 estimation, the rank decision problem, string pattern matching, sparse vector recovery, low-rank matrix and tensor recovery, and low-rank plus sparse matrix recovery.

A very recent work of [GKSY26] gives an almost-optimal $\Omega(\sqrt{n})$ lower bound on the white-box streaming space complexity of the *Longest Increasing Subsequence* (LIS) problem. Although near-tight deterministic lower bounds for LIS have been known for decades [GG10, EJ15], no superlogarithmic randomized streaming lower bound space is currently known.

Like the white-box model, the *pan-private* streaming model [DNP+10, MMNW11] also considers settings where the internal state of the streaming algorithm may be exposed. However, the two models differ in their objectives. The white-box model focuses on ensuring *correctness* in the face of adaptive adversaries, whereas the pan-private model is concerned with *privacy*. In the pan-private setting, stream elements typically represent sensitive individual data (*e.g.*, financial or medical records), and the goal is to ensure that even if the

internal state of the algorithm is compromised, the privacy of each individual's data remains protected.

2 Proof Overview

2.1 Lower Bound for Moment Estimation

We overview the proof of Theorem 1.1 in this section. For simplicity sake, we restrict our attention to the case of $p = 0$, where the F_p moment is simply the number of distinct elements in the stream and thus, estimating the F_p moment is the same as estimating the number of distinct elements. Essentially the same ideas also extend to other values of p, as streams with few distinct elements will have a lot of repetitions and therefore, a very different F_p moment than streams that have a lot of distinct elements.

As is standard, we prove the desired white-box streaming lower bound by proving a corresponding communication lower bound. For our result, we consider a k-party communication problem, called the Set Union (SU) problem, where all the parties $i \in [k]$ gets as input a set x_i of size t from a universe of size $n = 2kt$. The goal is to distinguish between the ("YES") case where the sets received by all the parties are the same implying that the size of the union of all the sets is small and the ("NO") case where the sets received by all the parties are (very) different implying that the size of the union of all the sets is large. As the size of the union is just the number of distinct elements, a lower bound for the Set Union problem also implies a lower bound for the problem of estimating the number of distinct elements.

In terms of parameters, we require that in the NO case, the size of the union of all the sets is at least $t + (k-1) \cdot t/2$, that is, on average every set other than x_1 contributes at least $t/2$ fresh elements to the union. As the size of the union in the YES case is t, showing that the YES and the NO cases are indistinguishable rules out the possibility of approximating the distinct elements to within a factor $\Omega(k)$. As k can be arbitrarily large, we have the desired result. We mention that the approximation factor we obtain for the k-party problem is tight up to constant factors. Indeed, it is easy to see that the size of the union lies in the interval $[t, kt]$ implying that a k-approximation is trivial. This also establishes that it is impossible to obtain Theorem 1.1 via two-party techniques (even for $p = 0$).

We show that a white-box communication protocol with communication $o(t)$ cannot distinguish between the YES case and the NO case of the Set Union problem. This is tight up to constant factors because of a simple protocol where the first party simply sends x_1 and all the other parties check if their sets are the same as x_1 or not.

White-box adversaries and a hybrid argument. We start by recalling the white-box communication model. In this model, the input x_1 of the first party is chosen by a (possibly randomized) adversary. Then, this party uses its own private randomness to compute a message m_1 to send to the second party. The adversary then sees this randomness and

chooses an input x_2 for the second party who can use it to compute a message m_2 using its own randomness. This continues till party k computes a message m_k that is either YES or NO, which constitutes the output of the protocol.

Note that, in the YES case of Set Union, all the adversary can do is to choose the input x_1 as the other inputs are promised to be the same. We call the set of all such adversaries $\mathbf{A}_1$. On the other hand, in the NO case, the adversary can choose any sets $x_1, \ldots, x_k$ as long as the promise $\left|\bigcup_{i\in[k]} x_i\right| \geq t + (k-1)\cdot t/2$ of a large union is satisfied. We call the set of all such adversaries $\mathbf{A}_k$. If there exists a white-box protocol π that distinguishes the YES case from the NO case, then π should output YES with high probability in the presence of any adversary from $\mathbf{A}_1$ and should output NO with high probability in the presence of any adversary from $\mathbf{A}_k$. In other words, if $q(\pi, \mathcal{A})$ denotes the probability that π outputs YES in the presence of adversary $\mathcal{A}$, the desired lower bound follows if we show that for any protocol π where every party sends $o(t)$ bits, it holds that (the first max in this expression can be replaced by a min. However, our proof establishes a stronger statement so we retain the max):

$$\max_{\mathcal{A}\in\mathbf{A}_1} q(\pi, \mathcal{A}) - \max_{\mathcal{A}\in\mathbf{A}_k} q(\pi, \mathcal{A}) = o(1). \tag{1}$$

To show this statement, we use the hybrid technique. Instead of arguing about adversaries from $\mathbf{A}_1$ and $\mathbf{A}_k$, we construct intermediate "hybrid" adversaries $\mathbf{A}_i$, for all $1 < i < k$, and show that the difference between consecutive hybrids is at most $o(1/k)$, that is, we have:

$$\max_{\mathcal{A}\in\mathbf{A}_i} q(\pi, \mathcal{A}) - \max_{\mathcal{A}\in\mathbf{A}_{i+1}} q(\pi, \mathcal{A}) = o(1/k). \tag{2}$$

By summing this result for all i, we have the desired lower bound. Roughly speaking, the set $\mathbf{A}_i$ is the set of adversaries that behave like an adversary from $\mathbf{A}_1$ for the first $k-i+1$ parties and an adversary from $\mathbf{A}_k$ for the last $i-1$ parties. That is, any adversary in the set $\mathbf{A}_i$ ensures that the input sets received by the first $k-i+1$ parties are always the same while the sets received by the last $i-1$ parties can be different but they satisfy the promise $\left|\bigcup_{i\in[k]} x_i\right| \geq t + (i-1)\cdot t/2$ so that each such set adds $t/2$ fresh elements to the union on average.

On hybrid mismatch. Before showing that the difference of outputting YES between pairs of consecutive hybrids is small, we note a subtle point about the max versus min issue mentioned above. For the purposes of our lower bound it suffices to show that Equation (1) with the max in the first term replaced by a min as a correct protocol should output YES with high probability for any adversary from the set $\mathbf{A}_1$. In terms of the proof, this means that while it suffices to show that *some* adversary in $\mathbf{A}_1$ is close to an adversary in $\mathbf{A}_k$, we actually show that *all* adversaries in $\mathbf{A}_1$ are close to an adversary in $\mathbf{A}_k$.

This difference turns out to be crucial for the hybrid argument to work. Indeed, suppose all we know is that *some* adversary in $\mathbf{A}_1$ is close to an adversary in $\mathbf{A}_2$ and *some* adversary in $\mathbf{A}_2$ is close to an adversary in $\mathbf{A}_3$ and so on, then there is no way to conclude anything about

the difference between $\mathbf{A}_1$ and $\mathbf{A}_3$ from just these claims. This is because the adversary in $\mathbf{A}_2$ that is close to $\mathbf{A}_1$ could be completely unrelated to the adversary in $\mathbf{A}_2$ that is close to $\mathbf{A}_3$. On the other hand, we have this claims about *all* adversaries, then it is straightforward to combine them and obtain a claim about closeness between $\mathbf{A}_1$ and $\mathbf{A}_3$. Put differently, the reason for proving Equation (1) with a max instead of min is to avoid a *mismatch* in the adversaries considered in different pairs of consecutive hybrids.

Showing consecutive hybrids are indistinguishable. It now boils down to showing that for protocols π with $o(t)$-communication, any pair i and $i+1$ of consecutive hybrids satisfies Equation (2). For this, note that both hybrids i and $i+1$ have the property that the input to the first $k-i$ parties is the same. Thus, for adversaries from both these hybrids, party $k-i$ knows that the input of any party before it matches his own, and can simulate their behavior. In turn, we can conclude that the first $k-i-1$ parties do not need to send anything, which effectively means we can ignore these parties in our analysis. This means that the proof for the case $i = k-1$, when no parties can be ignored, will also extend to all other i, and we can focus solely on this case.

To this end, fix an protocol π with $o(t)$ communication and look at Equation (2) for $i = k-1$. One of the main ingredients in our proof is that it is always possible to derandomize the first party. Indeed, note that if the first party uses randomness to compute its message it can go over all possible random strings and optimize to see which one performs the best over all the possible choices the adversary has for the future parties. The reason this is only true for the first party (and does not imply that the whole protocol can be derandomized which would have contradicted Theorem 1.4) is that the remaining parties do not know what inputs were given to the parties before them. All they see is the message they received which may not reveal the inputs entirely. Thus, they have no way of knowing the set of all possible choices the adversary has for the parties after them making it impossible to perform the optimization above.

Overall, we are now in situation where the first party is deterministic. Now, for each possible input x_1 for the first party, let $m_1 = m_1(x_1)$ be the message the first party sends to the second party. Note that the fact the π has low communication implies that the message m_1 is the same for many different inputs x_1 for the first party. Let S_1 be the set of all such inputs x_1 and consider what happens when the input to first party is replaced by a random input $x_1' \in S_1$ that may or may not be the same as x_1. With this modified input, if it is the case that the set of inputs to the parties satisfies the promise for $\mathbf{A}_{i+1}$, then the fact that the message m_1 is the same for both x_1 and x_1' implies that the protocol does not distinguish well between $\mathbf{A}_i$ and $\mathbf{A}_{i+1}$. Thus, it suffices to show that with high probability, a uniformly random input $x_1' \in S_1$ satisfies the promise for $\mathbf{A}_{i+1}$.

For this, recall that $i = k-1$ and note that the promise for $\mathbf{A}_{i+1}$ requires that $|x_1' \cup X_{>1}| \geq t + (k-1) \cdot t/2$, where we define $X_{>1} = \bigcup_{1<i\leq k} x_i$ for notational convenience. As any adversary in $\mathbf{A}_i$ requires that $x_1 = x_2$ and we have the promise for $\mathbf{A}_i$, we have that $|X_{>1}| \geq t + (k-2) \cdot t/2$. Thus, the only way the promise is violated is when $|x_1' \setminus X_{>1}| < t/2$. To

finish, we use the fact that x'_1 is sampled from a large set S_1 to get that the probability of this happening, even when conditioned on a given value of $X_{>1}$ that is determined by the parties' and the adversary's future randomness independently of the choice of x'_1, is very small, and in particular, is $o(1/k)$, as required for Equation (2).

2.2 Derandomization for Total Functions

We overview the proof of Theorem 1.3 in this section, focusing on the case $k = 3$, which can be easily generalized to any $k > 3$. Fix a 3-party white-box communication protocol π that computes a total function f. As mentioned in Section 2.1, the first party can easily be derandomized. Let $\mathsf{succ}_{x_1}(r_1)$ denote the maximum success probability of π in computing f, given that the first party's input is x_1 and its random string is r_1. In our derandomized protocol, like in the derandomized protocol of [ABJ+22], the first party fixes its random string to the string r_1 that maximizes $\mathsf{succ}_{x_1}(r_1)$.

Crucially, the first party can compute $\mathsf{succ}_{x_1}(r_1)$ by enumerating all possible white-box adversaries $\mathcal{A}$, calculating the success probability of π against each $\mathcal{A}$ conditioned on x_1 and r_1, and taking the minimum. To compute the success probability against a fixed $\mathcal{A}$, the first party iterates over all random strings for the second and third parties, simulates π, and compares the output to the correct value of f.

Like the first party, to derandomize its strategy, the second party aims to fix its randomness to the string r_2 that maximizes $\mathsf{succ}_{x_1,x_2}(r_2)$, the protocol's success probability given inputs x_1, x_2 and the random string r_2 for the second party (note that the first party is now deterministic). However, the second party does not know x_1; it only observes the message m_1 sent by the first party.

The key observation is that if π computes a total function, then for any x_3, $f(x_1, x_2, x_3) = f(x'_1, x_2, x_3)$, as long as on input x'_1 the first party also sends the message m_1. Since for a fixed r_2, π gives the same output on inputs (x_1, x_2, x_3) and (x'_1, x_2, x_3), and since $f(x_1, x_2, x_3) = f(x'_1, x_2, x_3)$, we get that the value of $\mathsf{succ}_{x_1,x_2}(r_2)$ only depends on the message m_1, and not on the specific value of x_1. Therefore, the second party can compute this value using any input x'_1 consistent with the received message m_1, instead of x_1.

We note that this argument *does not* extend to *partial* functions, since when computing $\mathsf{succ}_{x_1,x_2}(r_2)$, the second party must only consider adversaries $\mathcal{A}$ that generate inputs satisfying the promise of f. For example, suppose the promise of f requires that $x_1 = x_3$. If the second party substitutes the actual x_1 with some $x'_1 \neq x_1$ when simulating such an adversary, then the third party receives $x_3 = x'_1$, which differs from the true input x_1 held by the first party. As a result, the generated input tuple is "inconsistent", as it violates the promise of the function.

We note that in the proof of Theorem 1.1, we are able to avoid the consistency issue and successfully derandomize party $j = k - i$ by restricting our attention to adversaries that assign the same input to all of the first j parties. As a result, since party j knows its own input, it also knows the inputs of all earlier parties and therefore does not need to generate

inputs for them.

Observe that this issue of consistency does not arise in the case of total functions, where all input combinations are valid by definition, so any choice of x_1' remains consistent with any x_3.

In the case of two-party ($k = 2$) partial functions, no consistency issues arise, as there is no third party involved, and therefore no risk of violating the promise. This fact makes the derandomization of [ABJ+22] work. We note that their derandomization also requires f to be boolean (if not a total function).

2.3 Separation of White-Box from Deterministic

Theorem 1.3 and the derandomization result of [ABJ+22] leave open the possibility that partial functions or relations, $f(x_1, \ldots, x_k)$, that are either boolean with $k \geq 3$ or non-boolean with $k = 2$, might have white-box communication protocols that require significantly less memory than deterministic ones. Theorems 2.1 and 2.2 show that large separations are possible in both these cases. Note that Theorem 2.1 implies Theorem 1.4.

Theorem 2.1. *There is a boolean* partial *function $f(a, b, c)$, such that its white box 3-party communication complexity is $\tilde{O}(1)$, while its one-way deterministic communication complexity is $\tilde{\Omega}(n)$, where n is the length of the inputs. In fact, f also has a $\tilde{O}(1)$-memory white-box streaming algorithm.*

Roughly speaking, the inputs a, b and c to the partial function $f(a, b, c)$ are vectors of length n. It is promised that $a_i = b_i$ for at least 10% of the indices $i \in [n]$. Additionally, for every index i where $a_i \neq b_i$, it is promised that $c_i = \perp$. The goal is to distinguish between the following two cases: (i) For all indices i where $a_i = b_i$, we have $c_i = b_i$; or (ii) For all such indices i, we have $c_i \neq b_i$.

The following is a white box protocol for f: The second party sends to the third party a set of 100 random pairs (i, b_i) (no message from the first party to the second). The pairs are selected using *reservoir sampling*, which, upon receiving b_i, flips coins to decide whether to add (i, b_i) to the set. Note that had the protocol decided ahead of time on this random set of indices, the white-box adversary could have made sure that $a_i \neq b_i$ for all i's in the set. Next, the third party goes over all indices i in the set, searching for one with $c_i \neq \perp$. Such an index i should exist with high probability, and if $c_i = b_i$, the algorithm identifies case (i); otherwise, if $c_i \neq b_i$, it identifies case (ii).

To complete the proof of Theorem 2.1, we show an $\Omega(n)$ lower bound on the memory required by any deterministic algorithm that solves f. Observe that our derandomization argument from the proof of Theorem 1.3 fails when applied to the above white-box protocol for f. This is because c determines the set of indices where $a_i = b_i$. Therefore, without knowing this set, the second party cannot generate an input a' such that a' and b are consistent with the third party's input c.

Theorem 2.2. *There is a (non-boolean)* relation $g(a, b)$*, such that its white-box* 2*-party communication complexity is* $\tilde{O}(1)$*, while its one-way deterministic communication complexity is* $\tilde{\Omega}(n)$*, where* n *is the length of the inputs. In fact,* g *also has a* $\tilde{O}(1)$*-memory white-box streaming algorithm.*

Roughly speaking, the relation $g(a, b)$ considers input vectors a and b with the same promise as in f. The goal is to output a set of 100 indices such that, for at least one of them, $a_i = b_i$.

A white-box protocol for g simply has the second party output a uniformly random set of 100 coordinates (note that there is no communication between the parties). An $\Omega(n)$ deterministic lower bound for g is derived from that of f.

3 Model and Preliminaries

3.1 Notation

For integers $n, k \geq 1$, $[n]$ denotes $\{1, \ldots, n\}$, $[k, n]$ denotes $\{k, \ldots, n\}$, and $\binom{[n]}{k}$ represents the collection of all k-subsets of $[n]$. Throughout, logarithms are base-2.

A *total function* is a function that is defined for every input in its domain, whereas a *partial function* may only be defined on a subset of the inputs.

3.2 White-Box Communication

The standard approach of proving streaming lower bounds is via communication complexity. So we define the following (one-way) white-box communication model. Let π be a k-party protocol and $\mathcal{A}$ be an adversary. Initially, $m_0 = \perp$. At the i-th step, the i-th party computes m_i based on m_{i-1}, its input x_i, and its (private) randomness r_i[8], where $x_i \in X_i$ is picked by the adversary (possibly randomly) based on $m_1, \ldots, m_{i-1}, x_1, \ldots, x_{i-1}, r_1, \ldots, r_{i-1}$. At the end, the k-th party outputs m_k.

Let $f : X_1 \times \cdots \times X_k \to 2^Y$ be a relation. The *success probability of* π *in solving* f *against* $\mathcal{A}$ is defined to be $\Pr(m_k \in f(x_1, \ldots, x_k))$, where the probability is over both r_i and the randomness of $\mathcal{A}$, and $x_1, \ldots, x_k$ are the input generated by $\mathcal{A}$ on those randomness. Also define the communication complexity of π to be the *maximum* length of m_i over all $i \in [k]$ and all possible inputs $x_1, \ldots, x_k$. Note that the communication complexity is defined to be the maximum per-party communication due to the following connection between streaming and communication.

[8]The definition of the model can be modified to compute m_i based on m_{i-1}, x_i, and $r_1, \ldots, r_i$. This essentially corresponds to free access to a random oracle for streaming algorithms. Our lower bounds in this paper hold even against this strongest form of randomness. In particular, in the hybrid argument for proving Lemma 4.4, we partially derandomize a prefix of parties. This is essentially the same as revealing their randomness to all later parties.

Observation 3.1. *For any relation f, if there is a white-box streaming algorithm that solves f with space S, then there is also a white-box communication protocol (for any number of parties) that solves f with the same probability and communication S.*

4 Lower Bound for Moment Estimation

In this section, we prove Theorem 1.1, a white-box streaming lower bound for the moment estimation problem, defined as follows: Given a vector $u \in [n]^m$, we denote by f_i the frequency (number of appearances) of element $i \in [n]$ in u. The p-th moment of u is defined as $F_p(u) = \sum_{i\in[n]} f_i^p$. For $\alpha \geq 1$, an (insertion-only) streaming algorithm α-approximates F_p on $[n]$ with probability δ, if given a stream of elements $u = u_1, u_2, \ldots$, with $u_i \in [n]$, the algorithm outputs a number F such that $F_p(u) \leq F \leq \alpha F_p(u)$, with probability at least δ.

To prove Theorem 1.1, in Section 4.1, we introduce a k-party promise problem $\mathsf{SU}_{k,n,t,w}$ in the white-box communication model. A proof of the lower bound for $\mathsf{SU}_{k,n,t,w}$ is presented in Section 4.2.

4.1 k-Party Promise Problem $\mathsf{SU}_{k,n,t,w}$

Definition 4.1 (Set Union)**.** *For $k, n, t \geq 1$ and $w \in (0, 1)$, each party $i \in [k]$ has $x_i \in \binom{[n]}{t}$. $\mathsf{SU}_{k,n,t,w}$ needs to distinguish between the following two cases:*

(YES) $|\bigcup_{i\in[k]} x_i| = t$*, or equivalently* $x_1 = \cdots = x_k$*;*

(NO) $|\bigcup_{i\in[k]} x_i| \geq t + (k-1)tw$.

We remark that $\mathsf{SU}_{k,n,t,w}$ is similar to the communication problem used by [CK16], but our proof approach is quite different and tailored to white-box communication. To see the connection between $\mathsf{SU}_{k,n,t,w}$ and α-approximate F_p moment estimation, we have the following reductions.

Lemma 4.2. *For $p \in [0, 1)$ and $\alpha \geq 1$, if there is a white-box streaming algorithm that solves α-approximate F_p moment estimation on $[n]$ with space S, then there is also a white-box communication protocol that solves $\mathsf{SU}_{k,n,t,w}$ with the same probability and communication S, where $k = \lceil (2\alpha)^{1/(1-p)} \rceil, t = \lfloor n/(2k) \rfloor, w = 1/2$.*

Proof. In the YES case of $\mathsf{SU}_{k,n,t,w}$, the frequency vector has exactly t coordinates of value k and all other coordinates are zero. Its p-th moment is always $k^p t$. On the other hand, the frequency vector has at least $t + (k-1)tw > kt/2$ nonzero coordinates in the NO case. So the p-th moment is larger than $kt/2$. By Observation 3.1, a white-box streaming algorithm for α-approximate F_p moment estimation implies a white-box communication protocol for $\mathsf{SU}_{k,n,t,w}$ so long as $\alpha \cdot k^p t \leq kt/2$. This is satisfied by the given parameters[9]. □

[9]The reduction actually works for any value of t. $t = \lfloor n/(2k) \rfloor$ is needed only to satisfy the condition of Lemma 4.4. The same holds for Lemma 4.3.

Lemma 4.3. *For $p > 1$ and $\alpha \geq 1$, if there is a white-box streaming algorithm that solves α-approximate F_p moment estimation on $[n]$ with space S, then there is also a white-box communication protocol that solves $\mathsf{SU}_{k,n,t,w}$ with the same probability and communication S, where $k = \lceil (2\alpha)^{1/(p-1)} \rceil, t = \lfloor n/(2k) \rfloor, w = 1 - k^{-(p-1)}/2$.*

Proof. In the YES case of $\mathsf{SU}_{k,n,t,w}$, the frequency vector has exactly t coordinates of value k and all other coordinates are zero. Its p-th moment is always $k^p t$. On the other hand, the frequency vector has at least $t + (k-1)tw > ktw$ nonzero coordinates in the NO case. Meanwhile, all the coordinates sum up to kt and each coordinate is in $[k]$ because each element appears at most once in the input of each party. Note that $u^p + v^p \leq (u-1)^p + (v+1)^p$ for $1 < u \leq v < k$. It implies that the maximum possible value of the p-th moment is obtained when each nonzero coordinate is either 1 or k. As the number of coordinates with value k is less than $(kt - ktw)/(k-1) \leq 2t(1-w)$ due to $k \geq 2$, the p-th moment is upper bounded by $2k^p t(1-w) + kt = 2kt$. By Observation 3.1, a white-box streaming algorithm for α-approximate F_p moment estimation implies a white-box communication protocol for $\mathsf{SU}_{k,n,t,w}$ so long as $k^p t \geq \alpha \cdot 2kt$. This is satisfied by the given parameters. □

Now, we are ready to prove Theorem 1.1, assuming the following lower bound for $\mathsf{SU}_{k,n,t,w}$.

Lemma 4.4. *For $k, n, t \geq 1$ and $w \in (0,1)$ such that $2kt \leq n$, any white-box communication protocol that solves $\mathsf{SU}_{k,n,t,w}$ with constant probability better than $1/2$, requires $\Omega(t(1-w) - \log k)$ communication.*

Proof of Theorem 1.1. Suppose there is a white-box streaming algorithm for α-approximate F_p moment estimation on $[n]$ with space S and constant probability better than $1/2$. By Lemmas 4.2 and 4.3, there is also a white-box communication protocol for $\mathsf{SU}_{k,n,t,w}$ with communication S and the same probability, where $k = \lceil (2\alpha)^{1/|p-1|} \rceil, t = \lfloor n/(2k) \rfloor, w = 1 - k^{-\max(p-1,0)}/2$. Lemma 4.4 then implies that $S = \Omega(t(1-w) - \log k) = \Omega(n/k^{\max(p,1)} - O(\log n))$. Plugging in $k = O_p(\alpha^{1/|p-1|})$ concludes the proof. □

4.2 Proof of Lemma 4.4

This section constitutes a proof of Lemma 4.4. We first define the following sets of adversaries $\mathbf{A}_i$. For $i \in [k]$, an adversary is in $\mathbf{A}_i$ if for any protocol, the generated input satisfies all requirements below.

1. $x_1 = \cdots = x_{k-i+1}$.
2. $|\bigcup_{i'=1}^{k} x_{i'}| = |\bigcup_{i'=k-i+1}^{k} x_{i'}| \geq t + (i-1)tw$.
3. x_1 is uniformly random over $\binom{[n]}{t}$.

Intuitively, adversaries in $\mathbf{A}_i$ generate input such that the first $k-i+1$ parties get a (partial) YES case while the last i parties get a (partial) NO case. We say the input is of type i if the first two requirements above are satisfied.

For protocol π and adversary $\mathcal{A}$, let $q(\pi, \mathcal{A})$ denote the probability of π outputting YES against $\mathcal{A}$ (over the randomness of both π and $\mathcal{A}$)[10]. Also let $q_i(\pi) = \max_{\mathcal{A} \in \mathbf{A}_i} q(\pi, \mathcal{A})$ for $i \in [k]$. Fix a protocol π^* with communication C and probability $\delta > 1/2$. For any adversary $\mathcal{A} \in \mathbf{A}_1$, it only generates type-1 inputs, which are always in the YES case by Definition 4.1, so $q(\pi^*, \mathcal{A}) \geq \delta$. This further implies $q_1(\pi^*) \geq \delta$. Similarly, we have $q_k(\pi^*) \leq 1 - \delta$. Consider the maximization $Q = \max_\pi(q_1(\pi) - q_k(\pi))$ over all protocols π with (per-party) communication at most C. Altogether, this means $Q \geq q_1(\pi^*) - q_k(\pi^*) \geq 2\delta - 1 = \Omega(1)$.

On the other hand, however, we show that $Q = o(1)$ if $C = o(t(1-w) - O(\log n))$. This is a contradiction, hence proving Lemma 4.4. To this end, it is sufficient to show that $Q_i = \max_\pi(q_i(\pi) - q_{i+1}(\pi)) = o(1/k)$ for all $i \in [k-1]$, where the maximization is over all protocols π with communication at most C.

Fix i from now on. Let $j = k - i$. Let π'_i be a protocol with communication at most C such that $Q_i = q_i(\pi'_i) - q_{i+1}(\pi'_i)$. Observe that $q_i(\cdot)$ and $q_{i+1}(\cdot)$ only involve adversaries in $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$, who only generate type-i and type-$(i+1)$ inputs. Therefore, each of the first j parties of π'_i knows that $x_1 = \ldots = x_j$. In this case, the j-th party can simulate the first j parties and compute by itself the same message it would have in π'_i.

Let π''_i be the protocol obtained from π'_i by replacing the messages of the first $j-1$ parties by the empty message and having the j-th party simulate the first j parties to compute m_j. So we have $q_i(\pi''_i) - q_{i+1}(\pi''_i) = q_i(\pi'_i) - q_{i+1}(\pi'_i)$. Now, for any possible randomness r_j for the j-th party, let π''_{i,r_j} be the protocol obtained from π''_i by fixing the randomness of the j-th party to r_j. We select π_i to be π''_{i,r_j} for the r_j that maximizes $q_i(\pi''_{i,r_j}) - q_{i+1}(\pi''_{i,r_j})$. We next show that $q_i(\pi_i) - q_{i+1}(\pi_i) \geq q_i(\pi''_i) - q_{i+1}(\pi''_i) = q_i(\pi'_i) - q_{i+1}(\pi'_i) = Q_i$. As shown above, since we are considering only $q_i(\cdot)$ and $q_{i+1}(\cdot)$, we always that $x_1 = \ldots = x_j$, and thus it suffices for the adversary to only select inputs for parties $[j, k]$. Indeed, we have

$$\begin{aligned}
q_i(\pi''_i) - q_{i+1}(\pi''_i) &= \mathop{\mathbb{E}}_{r_j}[\max_{\mathcal{A} \in \mathbf{A}_i} q(\pi''_{i,r_j}, \mathcal{A})] - \mathop{\mathbb{E}}_{r_j}[\max_{\mathcal{A} \in \mathbf{A}_{i+1}} q(\pi''_{i,r_j}, \mathcal{A})] \\
&= \mathop{\mathbb{E}}_{r_j}[\max_{\mathcal{A} \in \mathbf{A}_i} q(\pi''_{i,r_j}, \mathcal{A}) - \max_{\mathcal{A} \in \mathbf{A}_{i+1}} q(\pi''_{i,r_j}, \mathcal{A})] \\
&= \mathop{\mathbb{E}}_{r_j}[q_i(\pi''_{i,r_j}) - q_{i+1}(\pi''_{i,r_j})].
\end{aligned}$$

So we get that $q_i(\pi_i) - q_{i+1}(\pi_i) \geq q_i(\pi''_i) - q_{i+1}(\pi''_i)$, as claimed.

Let $\mathcal{A}_i$ be an adversary in $\mathbf{A}_i$ such that $q_i(\pi_i) = q(\pi_i, \mathcal{A}_i)$. It suffices to construct another adversary $\mathcal{A}' \in \mathbf{A}_{i+1}$ such that $q(\pi_i, \mathcal{A}') \geq q(\pi_i, \mathcal{A}_i) - o(1/k)$. Let $M_j : x_j \mapsto m_j$ be the function computed by the j-th party in π_i[11]. Consider the adversary $\mathcal{A}' = \mathcal{A}'(\pi_i)$, who generates input $x'_1, \ldots, x'_k$ as follows:

1. Let $x_1 = \cdots = x_j$ be generated by $\mathcal{A}_i$. Sample x'_j uniformly at random from $Z_j(x_j) = \{x'_j \mid M_j(x'_j) = M_j(x_j)\}$. For the first j parties, generate $x'_1 = \cdots = x'_j$.

[10] Although $\mathsf{SU}_{k,n,t,w}$ is a promise problem, any protocol for it can be arbitrarily extended to also output YES/NO on input not in YES/NO cases.

[11] M_j depends solely on x_j, but not m_{j-1}, r_j, because the first $j-1$ parties do no computation and the j-th party is deterministic.

2. For $\ell \in [j+1, k]$, let x_ℓ be generated by $\mathcal{A}_i$, who pretends that for $i' \in [\ell-1]$, the i'-th party gets $x_{i'}$ as input[12]. If $|(\bigcup_{i'=j}^{\ell-1} x'_{i'}) \cup x_\ell| + (k-\ell)t \geq t + itw$, then generate $x'_\ell = x_\ell$ for the ℓ-th party. Otherwise, for *all* parties $i' \in [\ell, k]$, generate $x'_{i'}$ that is disjoint from $x'_1, \ldots, x'_{i'-1}$.

Remark 4.5. *Informally, $\mathcal{A}'$ faithfully simulates $\mathcal{A}_i$ except that:*

1. *It re-samples a uniformly random x'_j conditioned on $m_j = M_j(x_j)$.*
2. *Once $|(\bigcup_{i'=j}^{\ell-1} x'_{i'}) \cup x_\ell| + (k-\ell)t < t + itw$, it disregards $\mathcal{A}_i$ and generates disjoint sets for all remaining parties. This is well-defined because $2kt \leq n$.*

We first claim $\mathcal{A}' \in \mathbf{A}_{i+1}$. The first requirement is satisfied because $x'_1 = \cdots = x'_j$ and $j = k - (i+1) + 1$. For the second requirement, let $\ell^* \in [j, k]$ be the largest index such that $|(\bigcup_{i'=j}^{\ell^*-1} x'_{i'}) \cup x_{\ell^*}| + (k-\ell^*)t \geq t + itw$ during the execution[13]. By Item 2 of Remark 4.5, we have

$$|\bigcup_{i'=j}^{k} x'_{i'}| = |\bigcup_{i'=k-(i+1)+1}^{k} x'_{i'}| = |\bigcup_{i'=j}^{\ell^*} x'_{i'}| + (k-\ell^*)t \geq t + ((i+1)-1)tw.$$

So $\mathcal{A}'$ always generates type-$(i+1)$ input. Regarding the third requirement, as x_1 is uniformly random over $\binom{[n]}{t}$ due to $\mathcal{A}_i \in \mathbf{A}_i$, the probability of $\mathcal{A}'$ generating any x'_1 is

$$\sum_{x_1 : x'_1 \in Z_j(x_1)} \Pr(x_1) \cdot \frac{1}{|Z_j(x_1)|} = \sum_{x_1 \in Z_j(x'_1)} \Pr(x_1) \cdot \frac{1}{|Z_j(x'_1)|} = \frac{|Z_j(x'_1)|}{\binom{n}{t}} \cdot \frac{1}{|Z_j(x'_1)|} = \frac{1}{\binom{n}{t}}.$$

Therefore, $\mathcal{A}' \in \mathbf{A}_{i+1}$.

Finally, we conclude the proof of Lemma 4.4 by showing $q(\pi_i, \mathcal{A}') \geq q(\pi_i, \mathcal{A}_i) - o(1/k)$. Observe that by Item 1 of Remark 4.5, the resampling of $\mathcal{A}'$ ensures that m_j remains the same. As a result, if this *were* the only deviation from $\mathcal{A}_i$, i.e., $x'_\ell = x_\ell$ for all $\ell \in [j+1, k]$, the joint distribution of x_ℓ, r_ℓ, m_ℓ for $\ell \in [j+1, k]$ was exactly the same as if π_i *were* executed against $\mathcal{A}_i$. In particular, this means that the probability of outputting YES *would* be $q(\pi_i, \mathcal{A}_i)$. In the actual execution of π_i against $\mathcal{A}'$, this probability is lowered only in the case $|(\bigcup_{i'=j}^{\ell-1} x'_{i'}) \cup x_\ell| + (k-\ell)t < t + itw$ for some $\ell \in [j+1, k]$, which also implies $|x'_j \cup (\bigcup_{i'=j+1}^{k} x_{i'})| < t + itw$. Note that $|\bigcup_{i'=j+1}^{k} x_{i'}| = |\bigcup_{i'=k-i+1}^{k} x_{i'}| \geq t + (i-1)tw$ as $\mathcal{A}_i \in \mathbf{A}_i$ always generates type-i input. Combining the two inequalities, we get $|x'_j \setminus X'| < tw$, where $X' = \bigcup_{i'=j+1}^{k} x_{i'}$. Let $Z_j[m_j] = \{x_j \mid M_j(x_j) = m_j\}$. Altogether, we have

$$\begin{aligned}
&q(\pi_i, \mathcal{A}_i) - q(\pi_i, \mathcal{A}') \\
&\quad \leq \Pr(|x'_j \setminus X'| < tw) \\
&\quad = \sum_{m_j} \Pr(m_j) \cdot \sum_{X'} \Pr(X' \mid m_j) \cdot \Pr(|x'_j \setminus X'| < tw \mid m_j, X')
\end{aligned}$$

[12] In particular, this means $x_{j+1} = x_j = \cdots = x_1$.

[13] This is well defined because the inequality is always satisfied by index j.

$$= \sum_{m_j} \frac{|Z_j[m_j]|}{\binom{n}{t}} \cdot \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{x'_j \in Z_j[m_j]} \Pr(x'_j \mid m_j, X') \cdot \mathbb{1}[|x'_j \setminus X'| < tw]$$

(as x_j is uniformly random in $\binom{[n]}{t}$)

$$= \sum_{m_j} \frac{|Z_j[m_j]|}{\binom{n}{t}} \cdot \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{x'_j \in Z_j[m_j]} \frac{1}{|Z_j[m_j]|} \cdot \mathbb{1}[|x'_j \setminus X'| < tw]$$

(as x'_j is uniformly random in $Z_j[m_j]$ and independent of X' given m_j)

$$= \frac{1}{\binom{n}{t}} \cdot \sum_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{x'_j \in Z_j[m_j]} \mathbb{1}[|x'_j \setminus X'| < tw]$$

$$\leq \frac{1}{\binom{n}{t}} \cdot \sum_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{u=0}^{\lceil tw \rceil - 1} \binom{n - |X'|}{u} \binom{|X'|}{t-u}$$

(enumerating over $u = |x'_j \setminus X'|$)

$$\leq \frac{2^C}{\binom{n}{t}} \cdot \max_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{u=0}^{\lceil tw \rceil - 1} \binom{n - |X'|}{u} \binom{|X'|}{t-u}$$

(as π_i has communication at most C)

$$\leq \frac{2^C}{\binom{n}{t}} \cdot \max_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{u=0}^{\lceil tw \rceil - 1} \binom{n}{u} \binom{(k-1)t}{t-u} \quad (\text{as } |X'| \leq (k-j)t \leq (k-1)t)$$

$$\leq 2^C \cdot \max_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{u=0}^{\lceil tw \rceil - 1} \frac{\binom{(k-1)t}{t-u}}{\binom{n-u}{t-u}} \quad (\text{as } \tbinom{n}{t} = \tbinom{n}{u}\tbinom{n-u}{t-u})$$

$$= 2^C \cdot \max_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{u=0}^{\lceil tw \rceil - 1} \prod_{v=0}^{t-u-1} \frac{(k-1)t - v}{n - u - v}$$

$$\leq 2^C \cdot \max_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \sum_{u=0}^{\lceil tw \rceil - 1} \frac{1}{2^{t-u}} \quad (\text{as } n - u - v \geq 2kt - t \geq 2((k-1)t - v))$$

$$\leq 2^C \cdot \max_{m_j} \sum_{X'} \Pr(X' \mid m_j) \cdot \frac{2}{2^{t(1-w)}}$$

$$\leq 2^C \cdot \frac{2}{2^{t(1-w)}}$$

$$= o(\tfrac{1}{k}), \quad (\text{as } C = o(t(1-w) - \log k))$$

as claimed.

4.3 Proof of Theorem 1.2

This section proves Theorem 1.2, a white-box streaming space lower bound for estimating MMS. Fix $\alpha \geq 1$. An (insertion-only) streaming algorithm is said to α-approximate MMS with success probability δ if, on any stream of undirected edges $E = e_1, e_2, \ldots$, it outputs

a value M satisfying $\frac{1}{\alpha} MMS(E) \leq M \leq MMS(E)$ with probability at least δ, where $MMS(E)$ denotes the size of a maximum matching in the graph with edge set E.

Proof of Theorem 1.2. We show any white-box streaming algorithm P that α-approximates MMS with space s and success probability p implies a white-box algorithm P' that α-approximates F_0 on $[n]$ with space $s + O(\log T)$ and success probability p, where T is the length of the stream. The algorithm P' works by feeding P edges of a bipartite graph $G = (L \cup R, E)$, with $L = [n]$ and $R = [T]$ (assume that T is known to P' in advance), constructed as follows: P' maintains a counter i, initialized to 0. Upon receiving an item $a \in [n]$, it increments i and feeds the edge (a, i) to P. At the end of the stream, P' outputs the estimate produced by P multiplied by α.

Below we prove that for any stream A for P' (possibly obtained by a white-box adversary), the maximum matching size in the graph G, fed to P by P', equals the number of distinct elements in A, *i.e.*, $MMS(G) = F_0(A)$. Additionally, every white-box adversary $\mathcal{A}'$ for P' induces an adversary $\mathcal{A}$ for P: if, in time i, the adversary $\mathcal{A}'$ inserts the element $a \in [n]$ to the stream A, the adversary $\mathcal{A}$ inserts the edge (a, i). Since P is a white-box algorithm that α-approximates MMS with probability p, with probability at least p, it outputs an α-approximation of $MMS(G)$. Since $MMS(G) = F_0(A)$, this output, multiplied by α, is an α-approximation of $F_0(A)$. We got that, for every white-box adversary $\mathcal{A}'$, the algorithm P' outputs an α-approximation of $F_0(A)$ with probability p, and therefore P' is a white-box algorithm for α-approximation of $F_0(A)$ with probability p.

In terms of space, aside from the working memory of P, the algorithm P' only stores the counter $i \in [T]$, which requires $O(\log T)$ bits. Hence the total memory used by P' is at most $s + O(\log T)$. Observe that in the hard instances used to prove Theorem 1.1, the stream length satisfies $T = \Theta(n)$ and the lower bound follows even if the algorithm knows T. Therefore, by Theorem 1.1, every white-box streaming algorithm that α-approximates F_0 on streams of length $T = \Theta(n)$ over the universe $[n]$ with probability $p > 1/2$ must use $\Omega(n/\alpha - O(\log n))$ memory. Since P' uses at most $s + O(\log n)$ memory for streams of length n and succeeds with probability p, we conclude that if $p > 1/2$, then $s = \Omega(n/\alpha - O(\log n))$, as claimed by the theorem.

To conclude the proof, it remains to show that for every stream A, the graph G obtained from it satisfies, $MMS(G) = F_0(A)$. First, let M be any matching in G, and define $D := \{a \in [n] : \exists i \in [T] \text{ with } (a, i) \in M\}$. Since M is a matching, each left vertex $a \in [n]$ is incident to at most one edge of M, hence the mapping $(a, i) \mapsto a$ is injective on M and therefore $|M| = |D|$. Moreover, if $(a, i) \in M$ then by construction $a = a_i$, so every $a \in D$ appears in A, implying $|D| \leq F_0(A)$. Hence $|M| \leq F_0(A)$, and taking the maximum over matching gives $MMS(G) \leq F_0(A)$.

Conversely, let $D \subseteq [n]$ be the set of distinct elements appearing in A, so $|D| = F_0(A)$. For each $b \in D$, choose an (arbitrary) index $i(b) \in [T]$ such that $a_{i(b)} = b$; then $(b, i(b)) \in E$. The set $M := \{(b, i(b)) : b \in D\}$ is a matching: the vertices $b \neq b'$ correspond to different left vertices, therefore cannot be matched with the same right vertex. The reason is that each

right vertex $i \in [T]$ is incident to at most one edge in G (this edge connects it to the stream element at time i). Since $|M| = |D| = F_0(A)$, we get that $MMS(G) \geq F_0(A)$. Combining the two inequalities yields $MMS(G) = F_0(A)$, and the claim of the theorem follows.

□

5 Derandomization for Total Functions

This section constitutes a proof of Theorem 1.3, showing that every white-box communication protocol that computes a total function can be derandomized. Let $k \geq 1$, $X_1, \ldots, X_k, Y$ be sets, and $f : X_1 \times \cdots \times X_k \to Y$ be a total function. Fix a white-box communication protocol π with probability better than $1/2$. The proof is by induction on the parties. Suppose the first i parties are deterministic, which holds vacuously for $i = 0$. Let $M_i : (x_1, \ldots, x_i) \mapsto m_i$ be the (deterministic) function jointly computed by the first i parties. Also let $Z_i[m_i] = \{(x_1, \ldots, x_i) \mid M_i(x_1, \ldots, x_i) = m_i\}$. The $(i+1)$-th party is derandomized as follows.

1. Pick an arbitrary $(x'_1, \ldots, x'_i) \in Z_i[m_i]$.
2. Pick the best r_{i+1} that maximizes the probability of π (against the worst adversary), pretending that for $i' \in [i]$, the i'-th party gets x'_i as input. This is done by, for all possible r_{i+1} and adversary $\mathcal{A}$, computing the success probability of π, conditioned on r_{i+1}, against $\mathcal{A}$. This is well-defined because it can compute the success probability of π exactly given the input of the first i parties.

To show the correctness of the above derandomization, we first claim that the choice of $x'_1, \ldots, x'_i$ is irrelevant in the sense that f always evaluates to the same value.

Claim 5.1. *For $(x'_1, \ldots, x'_i), (x''_1, \ldots, x''_i) \in Z_i[m_i]$, and $x_{i'} \in X_{i'}$ for $i' \in [i+1, k]$, it holds that $f(x'_1, \ldots, x'_i, x_{i+1}, \ldots, x_k) = f(x''_1, \ldots, x''_i, x_{i+1}, \ldots, x_k)$.*

Proof. Consider two adversaries $\mathcal{A}'$ and $\mathcal{A}''$ that always generate $x'_1, \ldots, x'_i, x_{i+1}, \ldots, x_k$ and $x''_1, \ldots, x''_i, x_{i+1}, \ldots, x_k$, respectively. As the first i parties are deterministic, π always computes m_i against either $\mathcal{A}'$ or $\mathcal{A}''$. Furthermore, since both $\mathcal{A}'$ and $\mathcal{A}''$ always generate $x_{i+1}, \ldots, x_k$ for the remaining parties, the output distribution of π must be the same in two cases. Also note that π outputs $f(x'_1, \ldots, x'_i, x_{i+1}, \ldots, x_k)$ with probability more than $1/2$ against $\mathcal{A}'$ while outputting $f(x''_1, \ldots, x''_i, x_{i+1}, \ldots, x_k)$ with probability more than $1/2$ against $\mathcal{A}''$. This is possible only if $f(x'_1, \ldots, x'_i, x_{i+1}, \ldots, x_k) = f(x''_1, \ldots, x''_i, x_{i+1}, \ldots, x_k)$. □

Note that the assumption of the function being total is crucial as otherwise, $x'_1, \ldots, x'_i$ may not be compatible with $x_{i+1}, \ldots, x_k$. As a corollary of Claim 5.1, conditioned on m_i, the success probability of π is independent of both the actual input $x_1, \ldots, x_i$ for the first i parties and the choice of $x'_1, \ldots, x'_i$ made by the $(i+1)$-th party.

By arbitrarily fixing $x'_1, \ldots, x'_i$, the $(i+1)$-th party can then pick the best r_{i+1} to maximize the probability of π. Thus, this derandomization step never decreases the probability of π by

an averaging argument[14]. After derandomizing all k parties, we get a deterministic protocol that outputs correctly against all possible adversaries, implying that it must be correct on all possible inputs. This concludes the proof of Theorem 1.3 as our derandomization step does not increase communication.

6 Separation of White-Box from Deterministic

We use $\mathsf{ENE}_{n,t,w}$ defined in the following. Section 6.1 proves the optimal separation for 3-party boolean partial functions, using $\mathsf{ENE}_{n,t,w}$ with constant t, w, as claimed in Theorem 2.1.

Definition 6.1 (Equal Not Equal)**.** *For $n, t \geq 1$, $w \in (0,1)$, and $a_1, \ldots, a_n$, $b_1, \ldots, b_n$, $c_1, \ldots, c_n \in [0,t]$, let $I = \{i \mid a_i b_i \neq 0\}$. Define $\mathsf{ENE}_{n,t,w}(a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n)$ to be*

$$\begin{cases} 1, & \text{if } |I| \geq nw, \text{ and } \forall i \in I, c_i = b_i, \text{ and } \forall i \notin I, c_i = 0 \\ 0, & \text{if } |I| \geq nw, \text{ and } \forall i \in I, c_i \notin \{0, b_i\}, \text{ and } \forall i \notin I, c_i = 0 \\ \textit{undefined}, & \textit{otherwise} \end{cases}.$$

A similar separation is shown in the 2-party setting for relations in general, as claimed in Theorem 2.2. We use $\mathsf{hit}_{n,w}$ defined as follows. In Section 6.2, we derive the separation for $\mathsf{hit}_{n,w}$ as a corollary of the separation for $\mathsf{ENE}_{n,t,w}$.

Definition 6.2. *For $n \geq 1$, $w \in (0,1)$, and $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{0,1\}$, let $I = \{i | a_i b_i \neq 0\}$. A set $S \subset [n]$ of size at most $(100 \log n)/w$ is in the correct set of outputs for the instance $\mathsf{hit}_{n,w}(a_1, \ldots, a_n, b_1, \ldots, b_n)$ if and only if $|I| < nw$ or $S \cap I \neq \emptyset$.*

6.1 3-Party Boolean Partial Functions

The following lemma shows $\mathsf{ENE}_{n,t,w}$ can be efficiently computed by sampling in white-box streaming, and therefore also by a short 3-party communication protocol.

Lemma 6.3. *For $n, t \geq 1$ and $w \in (0,1)$, there is a white-box streaming algorithm that solves $\mathsf{ENE}_{n,t,w}$ with high probability in space $O((\log n \log nt)/w)$.*

Proof. The algorithm works as follows.

1. Ignore all a_i.
2. Independently sample and store each (i, b_i) with probability $(10 \log n)/(nw)$.

[14]This is specific to the white-box setting. In the white-box setting, the adversary picks the input x_{i+2} of the next party only after seeing the randomness r_{i+1} of the current party. So the success probability can be viewed as an expectation, over r_{i+1}, of a minimization, over x_{i+2}, of the conditional success probability. Thus, picking the best r_{i+1} is possible. In standard randomized setting, however, the entire input is fixed at the beginning. So it is not always possible to pick a single best r_{i+1} for all possible inputs.

3. Upon the first c_j such that $c_j \neq 0$ and (j, b_j) is stored, output $\mathbb{1}[c_j = b_j]$.

Let $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$ be generated by an arbitrary adversary. By Definition 6.1, $b_i, c_i \neq 0$ for all $i \in I$. Also note that for any j with $c_j \neq 0$, we know $j \in I$. So the algorithm never outputs incorrectly.

It now remains to bound the probability of the algorithm failing to output. Since $|I| \geq nw$, by Chernoff bound[15], at least one (i, b_i) for $i \in I$ is stored with high probability. If so, upon c_i for the first such i, the algorithm always outputs $\mathbb{1}[c_i = b_i]$ correctly.

Finally, again by Chernoff bound, the total number of (i, b_i) stored by the algorithm is $O((\log n)/w)$ with high probability. This concludes the proof as each (i, b_i) takes $O(\log n + \log t) = O(\log nt)$ space. □

In the rest of this section, we prove the deterministic lower bound for the corresponding 3-party communication problem, where Alice has $a_1, \ldots, a_n$, Bob has $b_1, \ldots, b_n$, and Charlie has $c_1, \ldots, c_n$.

First consider the following gap version of 2-party set disjointness.

Definition 6.4. *For $n \geq 1$ and $w \in (0, 1)$, Alice has $X_A \subseteq [n]$ and Bob has $X_B \subseteq [n]$. $\mathsf{SD}_{n,w}$ needs to decide if $X_A \cap X_B = \emptyset$ or $|X_A \cap X_B| \geq nw$.*

Lemma 6.5. *For $n \geq 1$ and $w \in (0, 1/4)$, any deterministic communication protocol (not necessarily one-way) for $\mathsf{SD}_{n,w}$ requires $\Omega((1/4 - w)^2 n)$ communication.*

Proof. The proof constructs a fooling set by probabilistic method. Consider a set of uniformly randomly sampled $(X, \overline{X})$, where X independently includes $i \in [n]$ with probability $1/2$. For each $(X, \overline{X})$ and $(X', \overline{X'})$ of the set, $|X \cap \overline{X'}| \geq nw$ and $|X' \cap \overline{X}| \geq nw$ with probability $1 - \exp(-\Omega((1/4 - w)^2 n))$ by Chernoff bound. On the other hand, $X \cap \overline{X} = \emptyset$ always holds. By union bound, a sampled set of size $\exp(\Omega((1/4 - w)^2 n))$ is a fooling set with nonzero probability. Thus, there is a fooling set of size $\exp(\Omega((1/4 - w)^2 n))$, concluding the proof. □

Using Lemma 6.5, we prove a lower bound for the following 3-party communication problem $\mathsf{3SI}_{n,t,w}$, which essentially reformulates the 3-party communication problem corresponding to $\mathsf{ENE}_{n,t,w}$. Indeed, Alice arbitrarily sets $a_1, \ldots, a_n \in [0, t]$ such that $a_i \neq 0$ if and only if $i \in X_A$. Meanwhile, Bob sets $b_1, \ldots, b_n$ such that $b_i = g_B(i)$ if $i \in X_B$ and $b_i = 0$ otherwise. Similarly, Charlie sets $c_1, \ldots, c_n$ such that $c_i = g_C(i)$ if $i \in X_C$ and $c_i = 0$ otherwise. It can be verified that the conditions and values of $\mathsf{ENE}_{n,t,w}$ and $\mathsf{3SI}$ are equivalent. For conciseness, we work with $\mathsf{3SI}_{n,t,w}$ in the following.

Definition 6.6. *For $n, t \geq 1$ and $w \in (0, 1)$, Alice has $X_A \subseteq [n]$, Bob has $X_B \subseteq [n]$ satisfying $|X_A \cap X_B| \geq nw$ and function $g_B : X_B \to [t]$, and Charlie has $X_C = X_A \cap X_B$ and function*

[15] Note that the existence of the adversary does not invalidate Chernoff bound. This is because the adversary has to generate the input b_i first, determining whether or not $i \in I$, and only after that will the algorithm sample. Formally, we can still define nw independent events $\mathcal{E}_{i'}$ denoting if the i'-th smallest index in I is sampled. All these events are also independent of the adversary.

$g_C : X_C \to [t]$. $\mathsf{3SI}_{n,t,w}$ *needs to decide if* $g_B(i) = g_C(i)$ *for all* $i \in X_C$, *or* $g_B(i) \neq g_C(i)$ *for all* $i \in X_C$.

Lemma 6.7. *There exist constants* $t > 1$ *and* $w \in (0, 1/4)$ *such that any one-way deterministic communication protocol for* $\mathsf{3SI}_{n,t,w}$ *requires* $\Omega(n)$ *communication.*

Let the constants t, w be determined later. Suppose for the sake of contradiction that there is a one-way deterministic communication protocol π_{3SI} for $\mathsf{3SI}_{n,t,w}$ with communication $C = o(n)$. Let m_A and m_B be the messages sent by Alice and Bob, respectively. Also let $Z_A[m_A]$ be the collection of X_A on which Alice sends m_A. Intuitively, without Alice sending a significant portion of X_A, Bob cannot even find any item in $X_A \cap X_B$. This is formalized by the following claim.

Claim 6.8. *For* $w \in (0, 1/4)$, *there exist* m_A, X_B *and* $z \in (0, w)$ *such that for all subset* $X' \subseteq X_B$ *of size at most* nz, $X_A \cap X' = \emptyset$ *for some* $X_A \in Z_A[m_A]$.

Proof. Fix w. Suppose for the sake of contradiction that for all m_A, X_B and $z \in (0, w)$, there exists subset $X' \subseteq X_B$ of size at most nz which satisfies $X_A \cap X' \neq \emptyset$ for any $X_A \in Z_A[m_A]$. Consider the following $\mathsf{SD}_{n,w}$ protocol π_{SD} (partially) simulating π_{3SI}.

1. Based on X_A, Alice sends m_A to Bob.
2. Based on m_A, X_B, Bob sends X' to Alice.
3. Alice outputs "$X_A \cap X_B = \emptyset$" if and only if $X_A \cap X' = \emptyset$.

Observe that if indeed $X_A \cap X_B = \emptyset$, then $X_A \cap X' = \emptyset$ regardless of X'. Otherwise, since $|X_A \cap X_B| \geq nw$, our above assumption on π_{3SI} implies $X_A \cap X' \neq \emptyset$. Therefore, π_{3SI} is a deterministic communication protocol for $\mathsf{SD}_{n,w}$ with communication

$$C + \log\left(\sum_{u=0}^{nz} \binom{n}{u}\right) \leq C + \log 2^{\mathbb{H}(z)n} = C + \mathbb{H}(z)n,$$

where $\mathbb{H}(z) = -z \log z - (1-z)\log(1-z)$. However, this contradicts Lemma 6.5 for sufficiently small z. □

Let $M_B : (m_A, X_B, g_B) \mapsto m_B$ be the function computed by Bob. Building upon Claim 6.8, we next show that Charlie cannot output correctly without Bob finding items in $X_A \cap X_B$.

Claim 6.9. *For fixed* m_A, X_B *and* $w, z \in (0, 1/4)$, *there exists* $t > 1/z$ *and* $g_B, g'_B : X_B \to [t]$ *such that* $|\{i \mid g_B(i) = g'_B(i)\}| \leq nz$ *and* $M_B(m_A, X_B, g_B) = M_B(m_A, X_B, g'_B)$.

Proof. The proof is via probabilistic method. Consider a set of $g_B : X_B \to [t]$ sampled uniformly at random. For each g_B, g'_B of the set, $|\{i \mid g_B(i) = g'_B(i)\}| \leq |X_B|z \leq nz$ with

probability $1-\exp(-\Omega((z-1/t)^2|X_B|)) \geq 1-\exp(-\Omega((z-1/t)^2 nw))$ by Chernoff bound. By union bound, there is a set of size $\exp(\Omega((z-1/t)^2 nw))$ such that every g_B, g'_B of the set satisfy $|\{i \mid g_B(i) = g'_B(i)\}| \leq nz$. Furthermore, since $C < \Omega((z-1/t)^2 nw)$ for sufficiently large t, there must be g_B, g'_B of the set satisfying $M_B(m_A, X_B, g_B) = M_B(m_A, X_B, g'_B)$ as well. □

Finally, we are ready to prove Lemma 6.7.

Proof of Lemma 6.7. Arbitrarily fix $w \in (0, 1/4)$. Let m_A, X_B and $z \in (0, w)$ be guaranteed by Claim 6.8. Also let $t > 1/z$ and $g_B, g'_B : X_B \to [t]$ be guaranteed by Claim 6.9. Then we have that $X' = \{i \mid g_B(i) = g'_B(i)\}$ is a subset of X_B and is of size at most nz. Claim 6.8 implies there exists $X_A \in Z_A[m_A]$ such that $X_A \cap X' = \emptyset$. Define $g_C(i) = g_B(i)$ for all $i \in X_C$. Observe that $g_C(i) \neq g'_B(i)$ for all $i \in X_C$ because $X' \cap X_C = X' \cap X_A \cap X_B = \emptyset$. In other words, π_{3SI} must output differently on $(X_A, (X_B, g_B), (X_C, g_C))$ and $(X_A, (X_B, g'_B), (X_C, g_C))$. However, Alice sends m_A in both cases, and Bob sends $M(m_A, X_B, g_B) = M(m_A, X_B, g'_B)$ in both cases by Claim 6.9. Consequently, Charlie cannot distinguish the two cases, contradicting the correctness of π_{3SI}. □

6.2 2-Party Relations

Note that a similar argument to Lemma 6.3 shows $\mathsf{hit}_{n,w}$ can be efficiently solved by sampling in white-box streaming, and therefore also by a short 2-party communication protocol.

Lemma 6.10. *For $n \geq 1$ and $w \in (0, 1)$, there is a white-box streaming algorithm for $\mathsf{hit}_{n,w}$ with high probability and space $O((\log^2 n)/w)$.*

Next, we show a deterministic lower bound for the 2-party communication problem corresponding to $\mathsf{hit}_{n,w}$, which is reformulated as $\mathsf{2SI}_{n,w}$.

Definition 6.11. *For $n \geq 1$ and $w \in (0, 1)$, Alice has $X_A \subseteq [n]$ and Bob has $X_B \subseteq [n]$. $\mathsf{2SI}_{n,w}$ needs to return $X' \subset [n]$ of size at most $(100 \log n)/w$ such that $X_A \cap X_B < nw$ or $X' \cap X_A \cap X_B \neq \emptyset$.*

Lemma 6.12. *There exists constant $w \in (0, 1/4)$ such that any one-way deterministic communication protocol for $\mathsf{2SI}_{n,w}$ requires $\Omega(n)$ communication.*

Proof. Suppose for the sake of contradiction that there is a protocol π_{2SI} for $\mathsf{2SI}_{n,w}$ with $o(n)$ communication. For any $t \geq 1$, a protocol π_{3SI} for $\mathsf{3SI}_{n,t,w}$ can be constructed by Alice and Bob simulating π_{2SI}, and Bob sending $(i, g_B(i))$ to Charlie for all $i \in X'$, which is sufficient for Charlie to decide. The communication of π_{3SI} is $o(n)$ as the size of X' is at most $(100 \log n)/w$. This contradicts Lemma 6.7.

□

References

[ABJ+22] Miklós Ajtai, Vladimir Braverman, T. S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. The white-box adversarial data stream model. In *International Conference on Management of Data (PODS)*, pages 15–27, 2022. 0, 1, 2, 3, 4, 5, 6, 10, 11

[AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In Philip N. Klein, editor, *Symposium on Discrete Algorithms (SODA)*, pages 1723–1742, 2017. 4

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. 2, 3

[BEJW+22] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, , and Eylon Yogev. A framework for adversarially robust streaming algorithms. *Journal of the ACM*, 69(2), 2022. 5, 6

[BJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004. 2

[BY20] Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *Symposium on Principlesvof Database Systems (PODS)*, pages 49–62, 2020. 5

[CK16] Amit Chakrabarti and Sagar Kale. Strong fooling sets for multi-player communication with applications to deterministic estimation of stream statistics. In *Symposium on Foundations of Computer Science (FOCS)*, pages 41–50, 2016. 2, 3, 4, 13

[CVM22] Sourav Chakraborty, N. V. Vinodchandran, and Kuldeep S. Meel. Distinct Elements in Streams: An Algorithm for the (Text) Book. In *European Symposium on Algorithms (ESA)*, pages 34:1–34:6, 2022. 3

[DNP+10] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *Innovations in Computer Science (ICS)*, pages 66–80, 2010. 6

[EJ15] Funda Ergün and Hossein Jowhari. On the monotonicity of a data stream. *Comb.*, 35(6):641–653, 2015. 6

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985. 3

[FW23] Ying Feng and David P. Woodruff. Improved algorithms for white-box adversarial streams. In *International Conference on Machine Learning (ICML)*, volume 202, pages 9962–9975, 2023. 2, 5, 6

[GG10] Anna Gál and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM Journal on Computing*, 39(8):3463–3479, 2010. 6

[GKSY26] Anna Gál, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Optimal white-box adversarial streaming lower bounds for approximating LIS length. In Shubhangi Saraf, editor, *17th Innovations in Theoretical Computer Science Conference, ITCS 2026, Bocconi University, Milan, Italy, January 27-30, 2026*, volume 362 of *LIPIcs*, pages 64:1–64:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. 6

[GLW+25] Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. Lifting linear sketches: Optimal bounds and adversarial robustness. In Michal Koucký and Nikhil Bansal, editors, *Symposium on Theory of Computing (STOC)*, pages 395–406, 2025. 6

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006. 2

[IW05] Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In Harold N. Gabow and Ronald Fagin, editors, *Symposium on Theory of Computing (STOC)*, pages 202–208, 2005. 2

[KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In Chandra Chekuri, editor, *Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014. 4

[MMNW11] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Symposium on Principles of Database Systems (PODS)*, pages 37–48, 2011. 6

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il