Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams[*]

Michael Kapralov[†] Jelani Nelson[‡] Jakub Pachocki[§] Zhengyu Wang[¶]
David P. Woodruff[||] Mobin Yahyazadeh[**]

April 6, 2017

Abstract

In the communication problem **UR** (universal relation) [KRW95], Alice and Bob respectively receive $x, y \in \{0,1\}^n$ with the promise that $x \neq y$. The last player to receive a message must output an index i such that $x_i \neq y_i$. We prove that the randomized one-way communication complexity of this problem in the public coin model is exactly $\Theta(\min\{n, \log(1/\delta)\log^2(\frac{n}{\log(1/\delta)})\})$ for failure probability δ. Our lower bound holds even if promised $support(y) \subset support(x)$. As a corollary, we obtain optimal lower bounds for ℓ_p-sampling in strict turnstile streams for $0 \leq p < 2$, as well as for the problem of finding duplicates in a stream. Our lower bounds do not need to use large weights, and hold even if promised $x \in \{0,1\}^n$ at all points in the stream.

We give two different proofs of our main result. The first proof demonstrates that any algorithm $\mathcal{A}$ solving sampling problems in turnstile streams in low memory can be used to encode subsets of $[n]$ of certain sizes into a number of bits below the information theoretic minimum. Our encoder makes adaptive queries to $\mathcal{A}$ throughout its execution, but done carefully so as to not violate correctness. This is accomplished by injecting random noise into the encoder's interactions with $\mathcal{A}$, which is loosely motivated by techniques in differential privacy. Our correctness analysis involves understanding the ability of $\mathcal{A}$ to correctly answer adaptive queries which have positive but bounded mutual information with $\mathcal{A}$'s internal randomness, and may be of independent interest in the newly emerging area of adaptive data analysis with a theoretical computer science lens. Our second proof is via a novel randomized reduction from Augmented Indexing [MNSW98] which needs to interact with $\mathcal{A}$ adaptively. To handle the adaptivity we identify certain likely interaction patterns and union bound over them to guarantee correct interaction on all of them. To guarantee correctness, it is important that the interaction hides some of its randomness from $\mathcal{A}$ in the reduction.

[*]This paper is a merger of [NPW17], and of work of Kapralov, Woodruff, and Yahyazadeh.

[†]EPFL. michael.kapralov@epfl.ch.

[‡]Harvard University. minilek@seas.harvard.edu. Supported by NSF grant IIS-1447471 and CAREER award CCF-1350670, ONR Young Investigator award N00014-15-1-2388, and a Google Faculty Research Award.

[§]OpenAI. jakub@openai.com. Work done while affiliated with Harvard University, under the support of ONR grant N00014-15-1-2388.

[¶]Harvard University. zhengyuwang@g.harvard.edu. Supported by NSF grant CCF-1350670.

[||]IBM Research Almaden. dpwoodru@us.ibm.com.

[**]Sharif University of Technology. mn.yahyazadeh@gmail.com. Work done while an intern at EPFL.

1 Introduction

In turnstile ℓ_0-sampling, a vector $z \in \mathbb{R}^n$ starts as the zero vector and receives coordinate-wise updates of the form "$z_i \leftarrow z_i + \Delta$" for $\Delta \in \{-M, -M+1, \ldots, M\}$. During a query, one must return a uniformly random element from $support(x) = \{i : z_i \neq 0\}$. The problem was first defined in [FIS08], where a data structure (or "sketch") for solving it was used to estimate the Euclidean minimum spanning tree, and to provide ε-approximations of a point set P in a geometric space (that is, one wants to maintain a subset $S \subset P$ such that for any set R in a family of bounded VC-dimension, such as the set of all axis-parallel rectangles, $||R \cap S|/|S| - |R \cap P|/|P|| < \varepsilon$). Sketches for ℓ_0-sampling were also used to solve various dynamic graph streaming problems in [AGM12a] and since then have been crucially used in almost all known dynamic graph streaming algorithms[1], such as for: connectivity, k-connectivity, bipartiteness, and minimum spanning tree [AGM12a], subgraph counting, minimum cut, and cut-sparsifier and spanner computation [AGM12b], spectral sparsifiers [AGM13], maximal matching [CCHM15], maximum matching [AGM12a, BS15, Kon15, AKLY16, CCE$^+$16, AKL17], vertex cover [CCHM15, CCE$^+$16], hitting set, b-matching, disjoint paths, k-colorable subgraph, and several other maximum subgraph problems [CCE$^+$16], densest subgraph [BHNT15, MTVV15, EHW16], vertex and hyperedge connectivity [GMT15], and graph degeneracy [FT16]. For an introduction to the power of ℓ_0-sketches in designing dynamic graph stream algorithms, see the recent survey of McGregor [McG14, Section 3]. Such sketches have also been used outside streaming, such as in distributed algorithms [HPP$^+$15, PRS16] and data structures for dynamic connectivity [KKM13, Wan15, GKKT15].

Given the rising importance of ℓ_0-sampling in algorithm design, a clear task is to understand the exact complexity of this problem. The work [JST11] gave an $\Omega(\log^2 n)$-bit space lower bound for data structures solving even the case $M = 1$ which fail with constant probability, and otherwise whose query responses are $(1/3)$-close to uniform in statistical distance. They also gave an upper bound for $M \leq \mathrm{poly}(n)$ with failure probability δ, which in fact gave $\min\{\|z\|_0, \Theta(\log(1/\delta))\}$ uniform samples from the support of z, using space $O(\log^2 n \log(1/\delta))$ (here $\|z\|_0$ denotes $|\,support(z)|$). Thus we say their data structure actually solves the harder problem of ℓ_0-sampling$_k$ for $k = \Theta(\log(1/\delta))$ with failure probability δ, where in ℓ_0-sampling$_k$ the goal is to recover $\min\{\|z\|_0, k\}$ uniformly random elements, without replacement, from $support(z)$. The upper and lower bounds in [JST11] thus match up to a constant factor for $k = 1$ and δ a constant. We note though in many settings, even if the final application desires constant failure probability, ℓ_0-sampling$_k$ with either failure probability $o(1)$ or $k > 1$ (or both) is needed as a subroutine (see Figure 1).

Universal relation. The work of [JST11] obtains its lower bound for ℓ_0-sampling (and some other problems) via reductions from *universal relation* (**UR**). This problem was defined in [KRW95] and arose in connection with the work [KW90] on circuit depth lower bounds. For $f : \{0,1\}^n \to \{0,1\}$, $D(f)$ is the minimum depth of a fan-in 2 circuit over basis $\{\neg, \vee, \wedge\}$ computing f. Meanwhile, $C(f)$ is defined as the minimum number of bits that need to be communicated in a correct deterministic protocol for Alice and Bob to solve the following communication problem: Alice receives $x \in f^{-1}(0)$ and Bob receives $y \in f^{-1}(1)$ (implying $x \neq y$), and they must agree on an index $i \in [n]$ such that $x_i \neq y_i$. It is shown in [KW90] that $D(f) = C(f)$, which they then used to show a tight $\Omega(\log^2 n)$ depth lower bound for monotone circuits solving undirected s-t connectivity. The work of [KRW95] then proposed a strategy to separate the complexity classes $\mathbf{NC}^1$ and $\mathbf{P}$: start with a function f on $\log n$ bits requiring depth $\Omega(\log n)$, then "compose" it with itself $k = \log n/\log\log n$ times (see

[1] The spectral sparsification algorithm of [KLM$^+$14] is a notable exception.

reference	problem	distribution	$k > 1$?	$\delta = o(1)$?
[FIS08]	Euclidean minimum spanning tree	ℓ_0	yes	
[AGM12a]	connectivity[2]	any		yes
[AGM12a]	k-connectivity[2]	any		yes
[AGM12a]	bipartiteness[2]	any		yes
[AGM12a]	minimum spanning tree	any		yes
[AGM12b]	subgraph counting	ℓ_0	yes	
[AGM12b]	minimum cut	any		yes
[AGM12b]	cut sparsifiers	any		yes
[AGM12b]	spanners	any	yes	yes
[AGM12b]	spectral sparsifiers	any		yes
[CCHM15]	maximal matching	ℓ_0	yes	yes
[BS15]	maximum matching (unweighted) maximum matching (weighted)	ℓ_0 ℓ_0	yes yes	 yes
[Kon15]	maximum matching	any	yes	yes
[AKLY16]	maximum matching	ℓ_0		yes
[AKL17]	maximum matching	ℓ_0		yes
[CCE+16]	maximum matching vertex cover hitting set b-matching disjoint paths k-colorable subgraph	ℓ_0		yes
[BHNT15]	densest subgraph	ℓ_0		yes
[MTVV15]	densest subgraph	ℓ_0	yes	yes
[EHW16]	densest subgraph	ℓ_0	yes	
[GMT15]	vertex connectivity hyperedge connectivity	any		yes
[FT16]	graph degeneracy	ℓ_0	yes	

Figure 1: Guarantees needed by various works using samplers as subroutines. The last two columns indicate whether the work needs to use a sampler that returns k samples at a time when queried for some $k > 1$, or for some subconstant failure probability δ even to achieve failure probability 1/3 in the main application. The "distribution" column indicates the output distribution needed from the sampler for the application ("any" means a support-finding subroutine is sufficient, i.e. it suffices for a query to return any index i for which $z_i \neq 0$).

[KW90] for the definition of composition). If one could prove a strong enough direct sum theorem for communication complexity after composition, even for a random f, this construction would yield a function that is provably in $\mathbf{P}$ (in fact even in $\mathbf{NC}^2$), but not $\mathbf{NC}^1$. Proving such a direct sum theorem is still open, and the statement that it is true is known as the "KRW conjecture"; see for example the recent works [GMWW14, DM16] on this conjecture. As a toy problem en route to resolving it, [KRW95] suggested proving a direct sum theorem for k-fold composition of a particular function $\mathbf{UR}$ that they defined. That task was positively resolved in [EIRS91] (see also [HW90]).

The problem $\mathbf{UR}$ abstracts away the function f, and Alice and Bob are simply given $x, y \in \{0,1\}^n$ with the promise that $x \neq y$. The players must then agree on any index i with $x_i \neq y_i$. The deterministic communication complexity of $\mathbf{UR}$ is nearly completely understood, with upper and lower bounds that match up to an additive 3 bits, even if one imposes an upper bound on the number of rounds of communication [TZ97]. Henceforth we also consider a generalized problem $\mathbf{UR}_k$, where the output must be $\min\{k, \|x-y\|_0\}$ distinct indices on which x, y differ. We also use $\mathbf{UR}^{\subset}, \mathbf{UR}_k^{\subset}$ to denote the variants when promised $\mathit{support}(y) \subset \mathit{support}(x)$, and also Bob knows $\|x\|_0$. Clearly $\mathbf{UR}, \mathbf{UR}_k$ can only be harder than $\mathbf{UR}^{\subset}, \mathbf{UR}_k^{\subset}$, respectively.

More than twenty years after its initial introduction in connection with circuit depth lower bounds, Jowhari et al. in [JST11] demonstrated the relevance of $\mathbf{UR}$ in the randomized one-way communication model for obtaining space lower bounds for certain streaming problems, such as various sampling problems and finding duplicates in streams. In the one-way version, Bob simply needs to find such an index i after a single message from Alice, and we only charge Alice's single message's length as the communication cost. If $\mathbf{R}_{\delta}^{\to,pub}(f)$ denotes the randomized one-way communication complexity of f in the public coin model with failure probability δ, [JST11] showed that the space complexity of $\mathsf{FindDuplicate}(n)$ with failure probability δ is at least $\mathbf{R}_{\frac{7}{8}+\frac{\delta}{8}}^{\to,pub}(\mathbf{UR})$. In $\mathsf{FindDuplicate}(n)$, one is given a length-$(n+1)$ stream of integers in $[n]$, and the algorithm must output some element $i \in [n]$ which appeared at least twice in the stream (note that at least one such element must exist, by the pigeonhole principle). The work [JST11] then showed a reduction demonstrating that any solution to ℓ_0-sampling with failure probability δ in turnstile streams immediately implies a solution to $\mathsf{FindDuplicate}(n)$ with failure probability at most $(1+\delta)/2$ in the same space (and thus the space must be at least $\mathbf{R}_{\frac{15}{16}+\frac{\delta}{16}}^{\to,pub}(\mathbf{UR})$). The same result is shown for ℓ_p-sampling for any $p > 0$, in which the output index should equal i with probability $|x_i|^p/(\sum_j |x_j|^p)$, and a similar result is shown even if the distribution on i only has to be close to this ℓ_p-distribution in variational distance (namely, the distance should be bounded away from 1). It is then shown in [JST11] that $\mathbf{R}_{\delta}^{\to,pub}(\mathbf{UR}) = \Omega(\log^2 n)$ for any δ bounded away from 1. The approach used though unfortunately does not provide an improved lower bound for $\delta \downarrow 0$.

Seemingly unnoticed in [JST11], we first point out here that the lower bound proof for $\mathbf{UR}$ in that work actually proves the same lower bound for the promise problem $\mathbf{UR}^{\subset}$. This observation has several advantages. First, it makes the reductions to the streaming problems trivial (they were already quite simple when reducing from $\mathbf{UR}$, but now they are even simpler). Second, a simple

[2] [AGM12a] describes these algorithms as only needing δ a constant, but for a different definition of support-finding: when the data structure fails, it should output Fail instead of behaving arbitrarily. They then cite [JST11] as providing the sampler they use, but unfortunately [JST11] does not solve this variant of this problem. This issue can be avoided by using [JST11] with $\delta < 1/\mathit{poly}(n)$ so that whp no failures occur throughout their algorithm, very slightly worsening their final space bound in the main application by a $\lg n$ factor. Alternatively, one could use another variant of sampling which we define in Section A.3 and which can be solved via a minor modification of [JST11], which allows retaining the bounds of [AGM12a] for connectivity, k-connectivity, and bipartiteness testing.

reduction from $\mathbf{UR}^{\subset}$ to sampling problems provides space lower bounds even in the strict turnstile model, and even for the simpler *support-finding* streaming problem for which when queried is allowed to return *any* element of *support*(z), without any requirement on the distribution of the index output. Both of these differences are important for the meaningfulness of the lower bound. This is because in dynamic graph streaming applications, typically z is indexed by $\binom{n}{2}$ for some graph on n vertices, and z_e is the number of copies of edge e in some underlying multigraph. Edges then are not deleted unless they had previously been inserted, thus only requiring correctness for strict turnstile streams. Also, for every single application mentioned in the first paragraph of Section 1 (except for the two applications in [FIS08]), the known algorithmic solutions which we cited as using ℓ_0-sampling as a subroutine actually only need a subroutine for the easier support-finding problem. Finally, third and most relevant to our current work's main focus, the straightforward reductions from $\mathbf{UR}^{\subset}$ to the streaming problems we are considering here do not suffer any increase in failure probability, allowing us to transfer lower bounds on $\mathbf{R}_{\delta}^{\rightarrow,pub}(\mathbf{UR}^{\subset})$ for small δ to lower bounds on various streaming problems for small δ. The work [JST11] could not provide lower bounds for the streaming problems considered there in terms of δ for small δ.

We now show simple reductions from $\mathbf{UR}^{\subset}$ to $\mathsf{FindDuplicate}(n)$ and from $\mathbf{UR}_k^{\subset}$ to support-finding$_k$. In support-finding$_k$ we must report $\min\{k, \|z\|_0\}$ elements in *support*(z). In the claims below, δ is the failure probability for the considered streaming problem.

Claim 1. *Any one-pass streaming algorithm for* $\mathsf{FindDuplicate}(n)$ *must use* $\mathbf{R}_{\delta}^{\rightarrow,pub}(\mathbf{UR}^{\subset})$ *space.*

Proof. We reduce from $\mathbf{UR}^{\subset}$. Suppose there were a space-S algorithm $\mathcal{A}$ for $\mathsf{FindDuplicate}(n)$. Alice creates a stream consisting of all elements of *support*(x) and runs $\mathcal{A}$ on those elements, then sends the memory contents of $\mathcal{A}$ to Bob. Bob then continues running $\mathcal{A}$ on $n+1-\|x\|_0$ arbitrarily chosen elements of $[n]\setminus$ *support*(y). Then there must be a duplicate in the resulting concatenated stream, i satisfies $x_i \neq y_i$ iff i is a duplicate. $\square$

Claim 2. *Any one-pass streaming algorithm for support-finding$_k$ in the strict turnstile model must use* $\mathbf{R}_{\delta}^{\rightarrow,pub}(\mathbf{UR}_k^{\subset})$ *bits of space, even if promised that* $z \in \{0,1\}^n$ *at all points in the stream.*

Proof. This is again via reduction from $\mathbf{UR}_k^{\subset}$. Let $\mathcal{A}$ be a space-S algorithm for support-finding$_k$ in the strict turnstile model. For each $i \in$ *support*(x), Alice sends the update $z_i \leftarrow z_i + 1$ to $\mathcal{A}$. Alice then sends the memory contents of $\mathcal{A}$ to Bob. Bob then for each $i \in$ *support*(y) sends the update $z_i \leftarrow z_i - 1$ to $\mathcal{A}$. Now note that z is exactly the indicator vector of the set $\{i : x_i \neq y_i\}$. $\square$

Claim 3. *Any one-pass streaming algorithm for ℓ_p-sampling for any $p \geq 0$ in the strict turnstile model must use* $\mathbf{R}_{\delta}^{\rightarrow,pub}(\mathbf{UR}_k^{\subset})$ *bits of space, even if promised* $z \in \{0,1\}^n$ *at all points in the stream.*

Proof. This is via straightforward reduction from support-finding$_k$, since reporting $\min\{k, \|z\|_0\}$ elements of *support*(z) satisfying some distributional requirements is only a harder problem than finding *any* $\min\{k, \|z\|_0\}$ elements of *support*(z). $\square$

The reductions above thus raise the question: what is the asymptotic behavior of $\mathbf{R}_{\delta}^{\rightarrow,pub}(\mathbf{UR}_k^{\subset})$?

Our main contribution: We prove for any δ bounded away from 1 and $k \in [n]$, $\mathbf{R}_{\delta}^{\rightarrow,pub}(\mathbf{UR}_k^{\subset}) = \Theta(\min\{n, t\log^2(n/t)\})$ where $t = \max\{k, \log(1/\delta)\}$. Given known upper bounds in [JST11], our lower bounds are optimal for $\mathsf{FindDuplicate}(n)$, support-finding, and ℓ_p-sampling for any $0 \leq p < 2$ for nearly the full range of n, δ (namely, for $\delta > 2^{-n^{.99}}$). Also given an upper bound of [JST11], our lower bound is optimal for ℓ_0-sampling$_k$ for nearly the full range of parameters n, k, δ (namely, for $t < n^{.99}$). Previously no lower bounds were known in terms of δ (or k). Our main theorem:

Theorem 1. *For any δ bounded away from 1 and $1 \le k \le n$, $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k^\subset) = \Theta(\min\{n, t\log^2(n/t)\})$.*

We give two different proofs of Theorem 1 (in Sections 3 and 4). Our upper bound is also new, though follows by minor modifications of the upper bound in [JST11] and thus we describe it in the appendix. The previous upper bound was $O(\min\{n, t\log^2 n\})$. We also mention here that it is known that the upper bound for both $\mathbf{UR}_k$ and ℓ_0-sampling$_k$ in two rounds (respectively, two passes) is only $O(t\log n)$ [JST11]. Thus, one cannot hope to extend our new lower bound to two or more passes, since it simply is not true.

1.1 Related work

The question of whether ℓ_0-sampling is possible in low memory in turnstile streams was first asked in [CMR05, FIS08]. The work [FIS08] applied ℓ_0-sampling as a subroutine in approximating the cost of the Euclidean minimum spanning tree of a subset S of a discrete geometric space subject to insertions and deletions. The algorithm given there used space $O(\log^3 n)$ bits to achieve failure probability $1/\mathrm{poly}(n)$ (though it is likely that the space could be improved to $O(\log^2 n \log\log n)$ with a worse failure probability, by replacing a subroutine used there with a more recent ℓ_0-estimation algorithm of [KNW10]). As mentioned, the currently best known upper bound solves ℓ_0-sampling$_k$ using $O(t\log^2 n)$ bits [JST11], which Theorem 1 shows is tight.

For ℓ_p-sampling, conditioned on not failing, the data structure should output i with probability $(1\pm\varepsilon)|x_i|^p/\|x\|_p^p$. The first work to realize its importance came even earlier than for ℓ_0-sampling: [CK04] showed that an ℓ_2-sampler using small memory would lead to a nearly space-optimal streaming algorithm for multiplicatively estimating $\|x\|_3$ in the turnstile model, but did not know how to implement such a data structure. The first implementation was given in [MW10], achieving space $\mathrm{poly}(\varepsilon^{-1}\log n)$ with $\delta = 1/\mathrm{poly}(n)$. . For $1 \le p \le 2$ the space was improved to $O(\varepsilon^{-p}\log^3 n)$ bits for constant δ [AKO11]. In [JST11] this bound was improved to $O(\varepsilon^{-\max\{1,p\}}\log(1/\delta)\log^2 n)$ bits for failure probability δ when $0 < p < 2$ and $p \ne 1$. For $p = 1$ the space bound achieved by [JST11] was a $\log(1/\varepsilon)$ factor worse: $O(\varepsilon^{-1}\log(1/\varepsilon)\log(1/\delta)\log^2 n)$ bits.

For finding a duplicate item in a stream, the question of whether a space-efficient randomized algorithm exists was asked in [Mut05, Tar07]. The question was positively resolved in [GR09], which gave an $O(\log^3 n)$-space algorithm with constant failure probability. An improved algorithm was given in [JST11], using $O(\log(1/\delta)\log^2 n)$ bits of space for failure probability δ.

2 Overview of techniques

We now describe our two proofs of Theorem 1. For the upper bound, [JST11] achieved $O(t\log^2 n)$, but in the appendix we show that slight modifications to their approach yield $O(\min\{n, t\log^2(n/t)\})$. Our main contribution is in proving an improved lower bound. Assume $t < cn$ for some sufficiently small constant c (since otherwise we already obtain an $\Omega(n)$ lower bound). In both our lower bound proofs in this regime, the proof is split into two parts: we show $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^\subset) = \Omega(\log\frac{1}{\delta}\log^2\frac{n}{\log\frac{1}{\delta}})$ and $\mathbf{R}_{.99}^{\to,pub}(\mathbf{UR}_k^\subset) = \Omega(k\log^2\frac{n}{k})$ separately. We give an overview the former here, which is the more technically challenging half. Our two proofs of the latter are in Sections 3.2 and 4.2.

2.1 Lower bound proof via encoding subsets and an adaptivity lemma

Our first proof of the lower bound on $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^\subset)$ is via an encoding argument. Fix m. A randomized encoder is given a set $S \subset [n]$ with $|S| = m$ and must output an encoding $\mathsf{ENC}(S)$, and a decoder sharing public randomness with the encoder must be able to recover S given only $\mathsf{ENC}(S)$. We consider such schemes in which the decoder must succeed with probability 1, and the encoding length is a random variable. Any such encoding must use $\Omega(\log\binom{n}{m}) = \Omega(m \log \frac{n}{m})$ bits in expectation for some S.

There is a natural, but sub-optimal approach to using a public-coin one-way protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$ to devise such an encoding/decoding scheme. The encoder pretends to be Alice with input x being the indicator set of S, then lets $\mathsf{ENC}(S)$ be the message M Alice would have sent to Bob. The decoder attempts to recover S by iteratively pretending to be Bob m times, initially pretending to have input $y = 0 \in \{0,1\}^n$, then iteratively adding elements found in S to y's support. Henceforth let $\mathbf{1}_T \in \{0,1\}^n$ denote the indicator vector of a set $T \subset [n]$.

Algorithm 1 Simple Decoder.

procedure DEC(M)
  $T \leftarrow \emptyset$
  for $r = 1, \ldots, m$ **do**
    Let i be Bob's output upon receiving message M from Alice when Bob's input is $\mathbf{1}_T$
    $T \leftarrow T \cup \{i\}$
  end for
  return T
end procedure

One might hope to say that if the original failure probability were $\delta < 1/m$, then by a union bound, with constant probability every iteration succeeds in finding a new element of S (or one could even first apply some error-correction to x so that the decoder could recover S even if only a constant fraction of iterations succeeded). The problem with such thinking though is that this decoder chooses y's adaptively! To be specific, $\mathcal{P}$ being a correct protocol means

$$\forall x, y \in \{0,1\}^n, \ \mathop{\mathbb{P}}_s(\mathcal{P} \text{ is correct on inputs } x, y) \geq 1 - \delta, \tag{1}$$

where s is the public random string that both Alice and Bob have access to. The issue is that even in the second iteration (when $r = 2$), Bob's "input" $\mathbf{1}_T$ *depends on* s, since T depends on the outcome of the first iteration! Thus the guarantee of (1) does not apply.

One way around the above issue is to realize that as long as every iteration succeeds, T is always a subset of S. Thus it suffices for the following event $\mathcal{E}$ to occur: $\forall T \subset S$, $\mathcal{P}$ is correct on inputs $\mathbf{1}_S, \mathbf{1}_T$. Then $\mathbb{P}_s(\neg\mathcal{E}) \leq 2^m \delta$ by a union bound, which is at most $1/2$ for $m = \lfloor \log_2(1/\delta) \rfloor - 1$. We have thus just shown that $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^\subset) = \Omega(\min\{n, \log\binom{n}{m}\}) = \Omega(\min\{n, \log\frac{1}{\delta}\log\frac{n}{\log(1/\delta)}\})$.

Our improvement is as follows. Our new decoder again iteratively tries to recover elements of S as before. We will give up though on having m iterations and hoping for all (or even most) of them to succeed. Instead, we will only have $R = \Theta(\log\frac{1}{\delta}\log\frac{n}{\log\frac{1}{\delta}})$ iterations, and our aim is for the decoder to succeed in finding a new element in S for at least a constant fraction of these R iterations. Simplifying things for a moment, let us pretend for now that all R iterations do succeed in finding a new element. $\mathsf{ENC}(S)$ will then be Alice's message M, together with the set $B \subset S$

of size $m - R$ not recovered during the R rounds, explicitly written using $\lceil \log \binom{n}{|B|} \rceil$ bits. If the decoder can then recover these R remaining elements, this then implies the decoder has recovered S, and thus we must have $|M| = \Omega(\log \binom{n}{m} - \log \binom{n}{|B|}) = \Omega(R \log \frac{n}{m})$. The decoder proceeds as follows. Just as before, initially the decoder starts with $T = \emptyset$ and lets i be the output of Bob on $\mathbf{1}_T$ and adds it to T. Then in iteration r, before proceeding to the next iteration, the decoder randomly picks some elements from B and adds them into T, so that the number of elements left to be uncovered is some fixed number n_r. These extra elements being added to T should be viewed as "random noise" to mask information about the random string s used by $\mathcal{P}$, an idea very loosely inspired by ideas in differential privacy. For intuition, as an example suppose the iteration $r = 1$ succeeds in finding some $i \in S$. If the decoder were then to add i to T, as well as $\approx m/2$ random elements from B to T, then the resulting T reveals only ≈ 1 bit of information about i (and hence about s). This is as opposed to the $\log m$ bits T could have revealed if the masking were not performed. Thus the next query in round $r = 2$, although correlated with s, has very weak correlation after masking and we thus might hope for it to succeed. This intuition is captured in the following lemma, which we prove in Section 3.1:

Lemma 1. *Consider $f\colon \{0,1\}^b \times \{0,1\}^q \to \{0,1\}$ and $X \in \{0,1\}^b$ uniformly random. If $\forall y \in \{0,1\}^q$, $\mathbb{P}(f(X,y) = 1) \le \delta$ where $0 < \delta < 1$, then for any random variable Y supported on $\{0,1\}^q$,*

$$\mathbb{P}(f(X,Y) = 1) \le \frac{I(X;Y) + H_2(\delta)}{\log \frac{1}{\delta}}, \tag{2}$$

where $I(X;Y)$ is the mutual information between X and Y, and H_2 is the binary entropy function.

Fix some $x \in \{0,1\}^n$. One should imagine here that $f(X,y)$ is 1 iff $\mathcal{P}$ fails when Alice has input x and Bob has input y in a $\mathbf{UR}^{\subset}$ instance, and the public random string is $X = s$. Then the lemma states that if $y = Y$ is not arbitrary, but rather random (and correlated with X), then the failure probability of the protocol is still bounded as long as the mutual information between X and Y is bounded. It is also not hard to see that this lemma is sharp up to small additive terms. Consider the case $x, y \in [n]$, and $f(x,y) = 1$ iff $x = y$. Then if X is uniform, for all y we have $\mathbb{P}(f(X,y) = 1) = 1/n$. Now consider the case where Y is random and equal to X with probability $t/\log n$ and is uniform in $[n]$ with probability $1 - t/\log n$. Then in expectation Y reveals t bits of X, so that $I(X;Y) = t$. It is also not hard to see that $\mathbb{P}(f(X,Y) = 1) \approx t/\log n + 1/n$.

In light of the strategy stated so far and Lemma 1, the path forward is clear: at each iteration r, we should add enough random masking elements to T to keep the mutual information between T and all previously added elements below, say, $\frac{1}{2} \log \frac{1}{\delta}$. Then we expect a constant fraction of iterations to succeed. The encoder knows which iterations do not succeed since it shares public randomness with the decoder (and can thus simulate it), so it can simply tell the decoder which rounds are the failed ones, then explicitly include in M correct new elements of S for the decoder to use in the place of Bob's wrong output in those rounds. A calculation shows that if one adds a $(1 - 1/K) \approx 2^{-1/K}$ fraction of the remaining items in S to T after drawing one more support element from Bob, the mutual information between the next query to Bob and the randomness used by $\mathcal{P}$ will be $O(K)$ (see Lemma 5). Thus we do this for K a sufficiently small constant times $\log \frac{1}{\delta}$. We will then have $n_r \approx (1 - 1/K)^r m$. Note that we cannot continue in this way once $n_r < K$ (since the number of "random noise" elements we inject should at least be one). Thus we are forced to stop after $R = \Theta(K \log(m/K)) = \Theta(\log \frac{1}{\delta} \log \frac{n}{\log \frac{1}{\delta}})$ iterations. We then set $m = \sqrt{n \log(1/\delta)}$, so that $\mathbf{R}_{\delta}^{\to,pub}(\mathbf{UR}^{\subset}) = \Omega(|R| \log \frac{n}{m}) = \Omega(\min\{n, \log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}}\})$ as desired.

The argument for lower bounding $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k^\subset)$ is a bit simpler, and in particular does not need rely on Lemma 1. Both the idea and rigorous argument can be found in Section 3.2, but again the idea is to use a protocol for this problem to encode appropriately sized subsets of $[n]$.

As mentioned above, our lower bounds use protocols for $\mathbf{UR}^\subset$ and $\mathbf{UR}_k^\subset$ to establish protocols for encoding subsets of some fixed size m of $[n]$. These encoders always consist of some message M Alice would have sent in a $\mathbf{UR}^\subset$ or $\mathbf{UR}_k^\subset$ protocol, together with a random subset $B \subset S$ (using $\lceil \log_2 |B| \rceil + \lceil \log \binom{n}{|B|} \rceil$ bits, to represent both $|B|$ and the set B itself). Here $|B|$ is a random variable. These encoders are thus *Las Vegas*: the length of the encoding is a random variable, but the encoder/decoder always succeed in compressing and recovering the subset. The final lower bounds then come from the following simple lemma, which follows from the source coding theorem.

Lemma 2. *Let s denote the number of bits used by the $\mathbf{UR}^\subset$ or $\mathbf{UR}_k^\subset$ protocol, and let s' denote the expected number of bits to represent B. Then $(1+\mathsf{s}+\mathsf{s}') \ge \log\binom{n}{m}$. In particular, $s \ge \log\binom{n}{m} - s' - 1$.*

Section 3.1 provides our first proof that $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^\subset) = \Omega(\min\{n, \log^2(\frac{n}{\log(1/\delta)}) \log \frac{1}{\delta}\})$. We extend our results in Section 3.2 to $\mathbf{UR}_k^\subset$ for $k \ge 1$, proving a lower bound of $\Omega(k \log^2(n/k))$ communication even for constant failure probability.

2.2 Lower bound proof via reduction from $\mathbf{AugIndex}_N$

Our second lower bound proof for $\mathbf{UR}^\subset$ is via a randomized reduction from $\mathbf{AugIndex}_N$ [MNSW98]. In this problem, Charlie receives $z \in \{0,1\}^N$ and Diane receives $j^* \in [N]$ together with z_j for $j = j^*+1, \ldots, N$, and Diane must output z_{j^*}. It is shown in [MNSW98] that $\mathbf{R}_\delta^{\to,pub}(\mathbf{AugIndex}_N) = \Omega(N)$ for any δ bounded away from 1/2. In our reduction, $N = \Theta(\log(1/\delta) \log^2 \frac{n}{\log(1/\delta)})$.

For $\mathbf{UR}^\subset$, we can also think of the problem as Alice being given $S \subseteq [n]$ and Bob being given $T \subsetneq S$, and Bob must output some element of $S \backslash T$. In $\mathbf{AugIndex}_N$, Charlie views his input as $L = \Theta(\log \frac{n}{\log(1/\delta)})$ blocks of bits of nearly equal size, where the ith block represents a subset S_i of $[u_i]$ in some collection $\mathcal{S}_{u_i,m}$ of sets, for some carefully chosen universe sizes u_i per block. Here $\mathcal{S}_{u_i,m}$ is a collection of subsets of $[u_i]$ of size m of maximal size such any two sets in the collection have intersection size strictly less than $m/2$. Furthermore, Diane's index j^* is in some particular block of bits corresponding to some set S_{i^*}, and Diane also knows S_i for $i > j$.

Now we explain the reduction. We assume some protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$, and we give a protocol $\mathcal{P}'$ for $\mathbf{AugIndex}_N$. First, we define the universe $A = \bigcup_{i=1}^L (\{i\} \times [u_i] \times [100^i])$, which has size n. Charlie then defines $S = \bigcup_{i=1}^L (\{i\} \times S_i \times [100^i])$. Charlie and Diane use public randomness to define a uniformly random permutation π on $[n]$. Charlie can compute $\pi(S)$. Also, since Diane knows S_i for $i > i^*$, she can define $T = \bigcup_{i=i^*+1}^L (\{i\} \times S_i \times [100^i])$ and compute $\pi(T)$. Then $\pi(S)$ and $\pi(T)$ are the inputs to Alice and Bob in the protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$. Charlie sends Diane the message Alice would have sent Bob in $\mathcal{P}$ if her input had been $\pi(S)$, and Diane simulates Bob to recover an element in $\pi(S) \backslash \pi(T)$. Importantly, Alice and Bob do not know anything about π at this point other than that $\pi(S) = S$ and $\pi(T) = T$. Thus, the protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$, if it succeeds, outputs an arbitrary element $j \in \pi(S) \backslash \pi(T)$, which is a deterministic function of the labels of elements in $\pi(S)$ and $\pi(T)$ and the randomness R that Alice and Bob share, which is independent from the randomness in π. Since π is still a uniformly random map conditioned on $\pi(S) = S$ and $\pi(t) = t$ for each $t \in T$, and $j \in \pi(S) \backslash \pi(T)$, it follows that $\pi^{-1}(t)$ is a uniformly random element of $S \backslash T$. After receiving $\pi^{-1}(j)$, if $(i, a, r) = \pi^{-1}(j)$, then Charlie and Diane reveal the pairs $((i, a, z), \pi((i, a, z)))$ for each $z \in [100^i]$ to Alice and Bob and Bob updates his set $\pi(T)$ to include $\pi(i, a, z)$ for each

$z \in [100^i]$. One can show that at each step in this process, if Alice and Bob succeed in outputting an arbitrary item j from $\pi(S) \setminus \pi(T)$, then this is a uniformly random item from $\pi(S) \setminus \pi(T)$. The fact that this item is uniformly random is crucial for arguing the number of computation paths of the protocol of Alice and Bob is $o(1/\delta)$ with good probability, over π, so that one can argue (see below) that with good probability on every such computation path Alice and Bob succeed on that path, over their randomness R. Although the idea of using a random permutation appeared in [JST11] to show that any public coin **UR** protocol can be made into one in which a uniformly random element of $S\backslash T$ is output, here we must use this idea adaptively, slowly revealing information about π and arguing that this property is maintained for each of Bob's successive queries.

Due to geometrically increasing repetitions of items for increasing i, a uniformly random element in $S\backslash T$ is roughly 100 times more likely to correspond to an item in S_{i^*} than in S_i for $i < i^*$. Thus if Diane simulates Bob to recover a random element in $S\backslash T$, it is most likely to recover an element j of S_{i^*}. She can then tell Bob to include $\pi(j)$ and its 100^{i^*} redundant copies to $\pi(T)$ and iterate.

There are several obstacles to overcome to make this work. First, iterating means using $\mathcal{P}$ adaptively, which was the same issue that arose in Section 2.1. Second, a constant fraction of the time (1/100), we expect to obtain an element *not* in S_{i^*}, but rather from some S_i for $i < i^*$. If this happened too often, then Diane would need to execute many queries to recover a sufficiently large number of elements from S_{i^*} in order to solve $\mathbf{AugIndex}_N$. This would then require a union bound over too many possible computation paths, which would not be possible as Alice likely would fail on one of them (over the choice of R). However, since the random permutation argument above ensures that at each step we receive a uniformly random item from the current set $S \setminus T$, if we continue for m iterations, we can argue that with large probability, our sequence of inputs T over the iterations with which Diane invokes Bob's output are all likely to come from a family $\mathcal{T}$ of size at most $2^{O(m)}$. Here we need to carefully construct this family to contain a smaller number of sets from levels i for which $i^* - i$ is larger so that the overall number of sets is small. Given this, we can union bound over all such T, for total failure probability $\delta|\mathcal{T}| \ll 1$. Furthermore, we can also argue that after m iterations, it is likely that we have recovered at least $m/2$ of the elements from S_{i^*}, which is enough to uniquely identify $S_{i^*} \in \mathcal{S}_{u_i,m}$ by the limited intersection property of $\mathcal{S}_{u_i,m}$.

3 Lower bounds via the adaptivity lemma

3.1 Communication Lower Bound for $\mathbf{UR}^{\subset}$

Consider a protocol $\mathcal{P}$ for $\mathbf{UR}^{\subset}$ with failure probability δ, operating in the one-way public coin model. When Alice's input is x and Bob's is y, Alice sends $\mathsf{Alice}(x)$ to Bob, and Bob outputs $\mathsf{Bob}(\mathsf{Alice}(x), y)$, which with probability at least $1-\delta$ is in *support*$(x-y)$. As mentioned in Section 2, we use $\mathcal{P}$ as a subroutine in a scheme for encoding/decoding elements of $\binom{[n]}{m}$ for $m = \lfloor \sqrt{n \log(1/\delta)} \rfloor$. We assume $\log \frac{1}{\delta} \leq n/64$, since for larger n we have an $\Omega(n)$ lower bound.

3.1.1 Encoding/decoding scheme

We now describe our encoding/decoding scheme $(\mathsf{ENC}, \mathsf{DEC})$ for elements in $\binom{[n]}{m}$, which uses $\mathcal{P}$ in a black-box way. The parameters shared by ENC and DEC are given in Algorithm 2.

As discussed in Section 2, on input $S \in \binom{[n]}{m}$, ENC computes $M \leftarrow \mathsf{Alice}(\mathbf{1}_S)$ as part of its output. ENC also outputs a subset $B \subseteq S$ computed as follows. Initially $B = S$ and $S_0 = S$. ENC proceeds in R rounds. In round $r \in [R]$, ENC computes $s_r \leftarrow \mathsf{Bob}(M, \mathbf{1}_{S\backslash S_{r-1}})$. Let $b \in \{0,1\}^R$

be such that b_r records whether Bob succeeds in round r. ENC also outputs b. If $s_r \in S_{r-1}$, i.e. $\mathsf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$ succeeds, ENC sets $b_r = 1$ and removes s_r from B (since the decoder can recover s_r from the $\mathbf{UR}^{\subset}$-protocol, ENC does not need to include it in B); otherwise ENC sets $b_r = 0$. At the end of round r, ENC picks a uniformly random set S_r in $\binom{S_{r-1} \setminus \{s_r\}}{n_r}$. In particular, ENC uses its shared randomness with DEC to generate S_r in such a way that ENC, DEC agree on the sets S_r (DEC will actually iteratively construct $C_r = S \setminus S_r$). We present ENC in Algorithm 3.

The decoding process is symmetric. Let $C_0 = \emptyset$ and $A = \emptyset$. DEC proceeds in R rounds. On round $r \in [R]$, DEC obtains $s_r \in S \setminus C_{r-1}$ by invoking $\mathsf{Bob}(M, \mathbf{1}_{C_{r-1}})$. By construction of C_{r-1} (to be described later), it is guaranteed that $S_{r-1} = S \setminus C_{r-1}$. Therefore, DEC recovers exactly the same s_r as ENC. DEC initially assigns $C_r \leftarrow C_{r-1}$. If $b_r = 1$, DEC adds s_r to both A and C_r. At the end of round r, DEC inserts many random items from B into C_r so that $C_r = S \setminus S_r$. DEC can achieve this because of the shared random permutation π when constructing S_r. In the end, DEC outputs $B \cup A$. We present DEC in Algorithm 4.

Algorithm 2 Variables shared by encoder ENC and decoder DEC.

$m \leftarrow \lfloor \sqrt{n \log \frac{1}{\delta}} \rfloor$
$K \leftarrow \lfloor \frac{1}{16} \log \frac{1}{\delta} \rfloor$
$R \leftarrow \lfloor K \log(m/4K) \rfloor$
for $r = 0, \ldots, R$ **do**
    $n_r \leftarrow \lfloor m \cdot 2^{-\frac{r}{K}} \rfloor$ ▷ $|S_r| = n_r$, and $\forall r\ n_r - n_{r+1} \geq 2$
end for
π is a random permutation on $[n]$ ▷ Used to generate S_r and C_r

Algorithm 3 Encoder ENC.

procedure ENC(S)
    $M \leftarrow \mathsf{Alice}(\mathbf{1}_S)$
    $A \leftarrow \emptyset$ ▷ the set DEC recovers just from M
    $S_0 \leftarrow S$ ▷ at end of round r, DEC still needs to recover S_r
    for $r = 1, \ldots, R$ **do**
        $s_r \leftarrow \mathsf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$ ▷ $s_r \overset{?}{\in} S_{r-1}$ found in round r
        $S_r \leftarrow S_{r-1}$
        if $s_r \in S_{r-1}$ **then** ▷ i.e. if s_r is a valid sample
            $b_r \leftarrow 1$ ▷ $b \in \{0,1\}^R$ indicating which rounds succeed
            $A \leftarrow A \cup \{s_r\}$
            $S_r \leftarrow S_r \setminus \{s_r\}$
        else
            $b_r \leftarrow 0$
        end if
        Remove $|S_r| - n_r$ elements from S_r with smallest π_a's among $a \in S_r$ ▷ now $|S_r| = n_r$
    end for
    return $(M,\ S \setminus A,\ b)$
end procedure

Algorithm 4 Decoder DEC.

procedure DEC(M, B, b)
  ▷ M is Alice($\mathbf{1}_S$)
  ▷ $b \in \{0,1\}^R$ indicates rounds in which Bob succeeds
  ▷ B contains all elements of S that DEC doesn't recover via M
  $A \leftarrow \emptyset$ ▷ the subset of S DEC recovers just from M
  $C_0 \leftarrow \emptyset$ ▷ subset of S we have built up so far
  for $r = 1, \ldots, R$ **do** ▷ each iteration tries to recover 1 element via M
    $C_r \leftarrow C_{r-1}$
    if $b_r = 1$ **then** ▷ this means Bob succeeds in round r
      $s_r \leftarrow \mathsf{Bob}(M, \mathbf{1}_{C_{r-1}})$ ▷ Invariant: $C_r = S \backslash S_r$ (S_r is defined in ENC)
      $A \leftarrow A \cup \{s_r\}$
      $C_r \leftarrow C_r \cup \{s_r\}$
    end if
    Insert $m - n_r - |C_r|$ items into C_r with smallest π_a's among $a \in B \backslash C_r$
      ▷ Random masking "Differential Privacy" step. Still n_r elements left to recover.
  end for
  return $B \cup A$
end procedure

3.1.2 Analysis

We have two random objects in our encoding/decoding scheme: (1) the random source used by $\mathcal{P}$, denoted by X, and (2) the random permutation π. These are independent.

First, we can prove that $\mathsf{DEC}(\mathsf{ENC}(S)) = S$. That is, for any fixing of the randomness in X and π, DEC will always decode S successfully. It is because ENC and DEC share X and π, so that DEC essentially simulates ENC. We formally prove this by induction in Lemma 3.

Now our goal is to prove that by using the $\mathbf{UR}^{\subset}$-protocol, the number of bits that ENC saves in expectation over the naive $\lceil \log \binom{n}{m} \rceil$-bit encoding is $\Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log(1/\delta)})$ bits. Intuitively, it is equivalent to prove the number of elements that ENC saves is $\Omega(\log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)})$. We formalize this in Lemma 4. Note that ENC also needs to output b (i.e., whether the Bob succeeds on R rounds), which takes R bits. By our setting of parameters, we can afford the loss of R bits. Thus it is sufficient to prove $\mathbb{E}\,|B| = |S| - \Omega(\log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)})$.

We have $|S| - |B| = \sum_{r=1}^{R} b_r$. In Lemma 1, we prove the probability that Bob fails on round r is upper bounded by $\frac{I(X;S_{r-1})+1}{\log \frac{1}{\delta}}$, where $I(X; S_{r-1})$ is the mutual information between X and S_{r-1}. Furthermore, we will show in Lemma 5 that $I(X; S_{r-1})$ is upper bounded by $O(K)$. By our setting of parameters, we have $\mathbb{E}\, b_r = \Omega(1)$ and thus $\mathbb{E}(|S| - |B|) = \Omega(R) = \Omega(\log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)})$.

Lemma 3. *DEC(ENC(S)) $= S$.*

Proof. We claim that for $r = 0, \ldots, R$, $\{S_r, C_r\}$ is a partition of S (S_r is defined in Algorithm 3, and C_r in Algorithm 4). We prove the claim by induction on r. Our base case is $r = 0$, for which the claim holds since $S_0 = S$, $C_0 = \emptyset$.

Assume the claim holds for $r - 1$ ($1 \le r \le R$), and we consider round r. On round r, by induction $S \backslash S_{r-1} = C_{r-1}$, the index s_r obtained by both ENC and DEC are the same. Initially

$S_r = S_{r-1}$ and $C_r = C_{r-1}$, and so $\{S_r, C_r\}$ is a partition of S. If s_r is a valid sample (i.e. $s_r \in S_{r-1}$), then $b_r = 1$, and ENC removes s_r from S_r and in the meanwhile DEC inserts s_r into C_r, so that $\{S_r, C_r\}$ remains a partition of S. Next, ENC repeats removing the a from S_r with the smallest π_a value until $|S_r| = n_r$. Symmetrically, DEC repeats inserting the a into C_r with the smallest π_a value among $a \in B \backslash C_r$, until $|C_r| = |S| - n_r$. In the end we have $|S_r| + |C_r| = |S|$, so ENC and DEC execute repetition the same number of times. Moreover, we can prove that during the same iteration of this repeated insertion, the element removed from S_r is exactly the same element inserted to C_r. This is because in the beginning of a repetition $\{S_r, C_r\}$ is a partition of S. We have $B \backslash C_r \subseteq S \backslash C_r = S_r$. Let a^* denote $a \in S_r$ that minimizes π_a. Then $a^* \in B \backslash C_r \subseteq S_r$ (since a^* will be removed from S_r, it has no chance to be included in S in ENC, so that B contains a^*), and π_{a^*} is also the smallest among $\{\pi_a : a \in B \backslash C_r\}$. Thus both ENC and DEC will take a^* (for ENC, to remove from S_r, and for DEC, to insert into C_r). Therefore, $\{S_r, C_r\}$ remains a partition of S.

Given the fact that $\{S_r, C_r\}$ is a partition of S, the s_r are the same in ENC and DEC. Furthermore, $A = \{s_r : b_r = 1, r = 1, \ldots, R\}$ are the same in ENC and DEC. We know $A \subseteq S$. Since ENC outputs $S \backslash A$, and DEC outputs $(S \backslash A) \cup A$, we have $\mathsf{DEC}(\mathsf{ENC}(S)) = S$. ◻

Lemma 4. *Let $W \in \mathbb{N}$ be a random variable with $W \leq m$ and $\mathbb{E}\, W \leq m - d$. Then $\mathbb{E}(\log \binom{n}{m} - \log \binom{n}{W}) \geq d \log(\frac{n}{m} - 1)$.*

Proof.

$$\begin{aligned}
\log \binom{n}{m} - \log \binom{n}{W} &= \log \frac{n!/(m!(n-m)!)}{n!/(W!(n-W)!)} \\
&= \sum_{i=1}^{m-W} \log \frac{n - W - i + 1}{m - i + 1} \\
&\geq (m - W) \cdot \log \frac{n - W}{m} \\
&\geq (m - W) \cdot \log \frac{n - m}{m}
\end{aligned}$$

Taking expectation on both sides, we have $\mathbb{E}(\log \binom{n}{m} - \log \binom{n}{W}) \geq d \log(\frac{n}{m} - 1)$. ◻

Lemma 1 (restated). Consider $f\colon \{0,1\}^b \times \{0,1\}^q \to \{0,1\}$ and $X \in \{0,1\}^b$ uniformly random. If $\forall y \in \{0,1\}^q$, $\mathbb{P}(f(X,y) = 1) \leq \delta$ where $0 < \delta < 1$, then for any r.v. Y supported on $\{0,1\}^q$,

$$\mathbb{P}(f(X,Y) = 1) \leq \frac{I(X;Y) + H_2(\delta)}{\log \frac{1}{\delta}},$$

where $I(X;Y)$ is the mutual information between X and Y, and H_2 is the binary entropy function.

Proof. It is equivalent to prove

$$I(X;Y) \geq \mathbb{E}(f(X,Y)) \cdot \log \frac{1}{\delta} - H_2(\delta).$$

By definition of mutual entropy $I(X;Y) = H(X) - H(X|Y)$, where $H(X) = b$ and we must show

$$H(X|Y) \leq H_2(\delta) + (1 - \mathbb{E}(f(X,Y))) \cdot b + \mathbb{E}(f(X,Y)) \cdot (b - \log \frac{1}{\delta}) = b + H_2(\delta) - \mathbb{E}(f(X,Y)) \cdot \log \frac{1}{\delta}.$$

The upper bound for $H(X|Y)$ is obtained by considering the following one-way communication problem: Alice knows both X and Y while Bob only knows Y, and Alice must send a single message to Bob so that Bob can recover X. The expected message length in an optimal protocol is exactly $H(X|Y)$. Thus, any protocol gives an upper bound for $H(X|Y)$, and we simply take the following protocol: Alice prepends a 1 bit to her message iff $f(X,Y)=1$ (taking $H_2(\delta)$ bits in expectation). Then if $f(X,Y)=0$, Alice sends X directly (taking b bits). Otherwise, when $f(X,Y)=1$, Alice sends the index of X in $\{x|f(x,Y)=1\}$ (taking $\log(\delta 2^b) = b - \log\frac{1}{\delta}$ bits). $\square$

Corollary 1. *Let X denote the random source used by the $\mathbf{UR}^{\subset}$-protocol with failure probability at most δ. If S is a fixed set and $T \subset S$, $\mathbb{P}(\mathsf{Bob}(\mathsf{Alice}(\mathbf{1}_S), \mathbf{1}_T) \notin S \backslash T) \leq \frac{I(X;T)+H_2(\delta)}{\log\frac{1}{\delta}}$.*

Lemma 5. *$I(X;S_r) \leq 6K$, for $r = 1,\ldots,R$.*

Proof. Note that $I(X;S_r) = H(S_r) - H(S_r|X)$. Since $|S_r| = n_r$ and $S_r \subseteq S$, $H(S_r) \leq \log\binom{m}{n_r}$. Here is the main idea to lower bound $H(S_r|X)$: By definition of conditional entropy, $H(S_r|X) = \sum_x p_x \cdot H(S_r|X=x)$. We fix an arbitrary x. If we can prove that for any $T \subseteq S$ where $|T| = n_r$, $\mathbb{P}(S_r = T|X=x) \leq p$, then by definition of entropy we have $H(S_r|X=x) \geq \log\frac{1}{p}$.

First we can prove for any fixed T,

$$\mathbb{P}(S_r = T|X=x) \leq \prod_{i=1}^{r} \frac{\binom{n_{i-1}-n_r-1}{n_{i-1}-n_i-1}}{\binom{n_{i-1}-1}{n_{i-1}-n_i-1}}. \tag{3}$$

We have $\mathbb{P}(S_r = T|X=x) = \Pi_{i=1}^{r}\mathbb{P}(T \subseteq S_i | T \subseteq S_{i-1})$. On round i ($1 \leq i \leq r$), ENC removes $n_{i-1} - n_i$ elements (at least $n_{i-1} - n_i - 1$ of which are chosen all at random) from S_{i-1} to obtain S_i. Conditioned on the event that $T \subseteq S_{i-1}$, the probability that $T \subseteq S_i$ is at most $\binom{n_{i-1}-n_r-1}{n_{i-1}-n_i-1}/\binom{n_{i-1}-1}{n_{i-1}-n_i-1}$, where the equation achieves when $s_i \in S_{i-1}\backslash T$, and ENC takes a uniformly random subset of $S_{i-1}\backslash\{s_i\}$ of size $n_{i-1} - n_i - 1$, so that the subset does not intersect with T.

Next we can prove

$$\prod_{i=1}^{r} \frac{\binom{n_{i-1}-n_r-1}{n_{i-1}-n_i-1}}{\binom{n_{i-1}-1}{n_{i-1}-n_i-1}} \leq \frac{2^{6K}}{\binom{m}{n_r}}. \tag{4}$$

For notational simplicity, let $n^{\underline{k}}$ denote $n \cdot (n-1) \ldots (n-k+1)$. We have

$$\prod_{i=1}^{r} \frac{\binom{n_{i-1}-n_r-1}{n_{i-1}-n_i-1}}{\binom{n_{i-1}-1}{n_{i-1}-n_i-1}} = \prod_{i=1}^{r} \frac{(n_{i-1}-n_r-1)!n_i!}{(n_{i-1}-1)!(n_i-n_r)!} = \prod_{i=1}^{r} \frac{n_i^{\underline{n_r}}}{(n_{i-1}-1)^{\underline{n_r}}} = \prod_{i=1}^{r} \left(\frac{n_i^{\underline{n_r}}}{n_{i-1}^{\underline{n_r}}} \cdot \frac{n_{i-1}}{n_{i-1}-n_r} \right). \tag{5}$$

By telescoping,

$$\prod_{i=1}^{r} \frac{n_i^{\underline{n_r}}}{n_{i-1}^{\underline{n_r}}} = \frac{n_r^{\underline{n_r}}}{n_0^{\underline{n_r}}} = \frac{n_r!(n_0-n_r)!}{n_0!} = \frac{1}{\binom{n_0}{n_r}} = \frac{1}{\binom{m}{n_r}}. \tag{6}$$

Moreover,

$$\prod_{i=1}^{r} \frac{n_{i-1}}{n_{i-1}-n_r} \leq \prod_{i=1}^{r} \frac{1}{1-\frac{m\cdot 2^{-r/K}}{m\cdot 2^{-(i-1)/K}-1}} \leq \prod_{i=1}^{r} \frac{1}{1-\frac{m\cdot 2^{-r/K}+1}{m\cdot 2^{-(i-1)/K}}} = \prod_{j=1}^{r} \frac{1}{1-2^{-j/K}-\frac{2^{\frac{r-j}{K}}}{m}}. \tag{7}$$

By our setting of parameters

$$\frac{2^{\frac{r}{K}}}{m} \leq \frac{2^{\frac{R}{K}}}{m} \leq \frac{1}{4K}.$$

Therefore, for $j \in \{1, \ldots, r\}$,

$$\frac{1}{1-2^{-\frac{j}{K}}-\frac{2^{\frac{r-j}{K}}}{m}} \leq \frac{1}{1-(1+\frac{1}{4K})2^{-\frac{j}{K}}}.$$

By Taylor series $2^{1/K} = \sum_{n=0}^{\infty} \frac{(\ln 2)^n}{n! K^n} > 1 + \frac{\ln 2}{K} > 1 + \frac{1}{4K}$, and thus $\frac{1}{1-(1+\frac{1}{4K})2^{-j/K}} \leq \frac{1}{1-2^{(1-j)/K}}$, for $j = 2, \ldots, r$. For $j = 1$, we have $\frac{1}{1-(1+\frac{1}{4K})2^{-\frac{1}{K}}} \leq 2^K$.

By Lemma 6, we have $\prod_{j=1}^{\infty} \frac{1}{1-2^{-j/K}} \leq 2^{5K}$. Therefore, the right hand side of (7) is upper bounded by 2^{6K}. Together with (6), we prove (4) holds.

Finally, let $p = 2^{6K}/\binom{m}{n_r}$, we have $\mathbb{P}(S_r = T|X = x) \leq p$ and thus $H(S_r|X = x) \geq \log \frac{1}{p} = \log \binom{m}{n_r} - 6K$. Therefore, $H(S_r|X) \geq \log \binom{m}{n_r} - 6K$ and so $I(X; S_r) = H(S_r) - H(S_r|X) \leq 6K$. $\square$

Lemma 6. *Let $K \in \mathbb{N}$ and $K \geq 1$. We have $\prod_{j=1}^{\infty} \frac{1}{1-2^{-j/K}} \leq 2^{5K}$.*

Proof. First, we bound the product of first $2K$ terms. Note that $\frac{1}{1-2^{-x}} \leq \frac{8}{3x}$ for $0 < x \leq 2$. Therefore,

$$\prod_{j=1}^{2K} \frac{1}{1-2^{-j/K}} \leq (8/3)^{2K} \cdot \frac{K^{2K}}{(2K)!} \leq (8/3)^{2K} \cdot \frac{K^{2K}}{(2K/e)^{2K}} = (4e/3)^{2K} < 2^{4K}. \tag{8}$$

Then, we bound the product of the rest terms

$$\prod_{j=2K+1}^{\infty} \frac{1}{1-2^{-j/K}} \leq \prod_{j=2K+1}^{\infty} \frac{1}{1-2^{-\lfloor j/K \rfloor}} \leq \prod_{i=2}^{\infty} \left(\frac{1}{1-2^{-i}}\right)^K \leq \left(\frac{1}{1-\sum_{i=2}^{\infty} 2^{-i}}\right)^K = 2^K. \tag{9}$$

Multiplying two parts proves the lemma. $\square$

Theorem 2. $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^{\subset}) = \Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log(1/\delta)})$*, given that* $64 \leq \log \frac{1}{\delta} \leq \frac{n}{64}$.

Proof. By Lemma 3, the success probability of protocol $(\mathsf{ENC}, \mathsf{DEC})$ is 1. By Lemma 2, we have $\mathsf{s} \geq \log\binom{n}{m} - \mathsf{s}' - 1$, where $\mathsf{s}' = \log n + R + \mathbb{E}(\log\binom{n}{|B|})$. The size of B is $|B| = |S| - \sum_{r=1}^{R} b_r$. By Corollary 1, conditioned on S, $\mathbb{P}(b_r = 0) \leq \frac{I(X;S_{r-1})+1}{\log \frac{1}{\delta}}$. By Lemma 5, $I(X; S_{r-1}) \leq 6K$ (Note that when $r = 1$, $I(X; S_0) = 0 \leq 6K$). Therefore, $\mathbb{E}(b_r) \geq 1 - \frac{6K+1}{\log \frac{1}{\delta}}$. By the setting of parameters (see Algorithm 2) we have $\mathbb{E}(b_r) \geq \frac{39}{64}$. Therefore, $\mathbb{E}(|B|) \leq |S| - \frac{39}{64}R$. By Lemma 4, $\log\binom{n}{m} - \mathbb{E}(\log\binom{n}{|B|}) \geq \frac{39}{64}R \cdot \log(\frac{n}{m} - 1) \geq \frac{1}{2}R\log(\frac{n}{\log(1/\delta)})$. Furthermore, $\frac{1}{6}R\log\frac{n}{\log(1/\delta)} \geq R$. Thus we obtain $\mathsf{s} \geq \frac{R}{3}\log\frac{n}{\log(1/\delta)} - (\log n + 1) = \Omega(\log\frac{1}{\delta}\log^2\frac{n}{\log(1/\delta)})$. $\square$

3.2 Communication Lower Bound for $\mathbf{UR}_k^{\subset}$

In this section, we prove the lower bound $\mathbf{R}_{1/2}^{\to,pub}(\mathbf{UR}_k^{\subset}) = \Omega(\min\{n, k\log^2 \frac{n}{k}\})$. In fact, our lower bound holds for any failure probability δ bounded away from 1. Let $\mathcal{P}$ denote a $\mathbf{UR}_k^{\subset}$-protocol where Alice sends $\mathsf{Alice}_k(x)$ to Bob, and Bob outputs $\mathsf{Bob}_k(\mathsf{Alice}_k(x), y)$. We consider the following encoding/decoding scheme $(\mathsf{ENC}_k, \mathsf{DEC}_k)$ for $S \in \binom{[n]}{m}$. ENC_k computes $M \leftarrow \mathsf{Alice}_k(\mathbf{1}_S)$ as part of its message. In addition, ENC_k includes $B \subseteq S$ constructed as follows, spending $\lceil \log \binom{n}{|B|} \rceil$ bits. Initially $B = S$, and ENC_k proceeds in $R = \Theta(\log(n/k))$ rounds. Let $S_0 = S \supseteq S_1 \supseteq \ldots \supseteq S_R$ where S_r is generated by sub-sampling each element in S_{r-1} with probability $\frac{1}{2}$. In round r ($r = 1, \ldots, R$), ENC_k tries to obtain k elements from S_{r-1} by invoking $\mathsf{Bob}_k(M, \mathbf{1}_{S \setminus S_{r-1}})$, denoted by A_k, and removes $A_k \cap (S_{r-1} \setminus S_r)$ (whose expected size is $\frac{k}{2}$) from B. Note that DEC_k is able to recover the elements in $A_k \cap (S_{r-1} \setminus S_r)$. For each round the failure probability of Bob_k is at most δ. Thus we have $\mathbb{E}(|S| - |B|) \geq \frac{k}{2} \cdot (1-\delta) \cdot R = \Omega(k \log \frac{n}{k})$. Furthermore, each element contains $\Theta(\log \frac{n}{k})$ bits of information, thus yielding a lower bound of $\Omega(k \log^2 \frac{n}{k})$ bits.

In this section we assume $k \leq n/2^{10}$, since for larger n we have an $\Omega(n)$ lower bound.

3.2.1 Encoding/decoding scheme

Algorithm 5 Variables Shared by Encoder ENC_k and Decoder DEC_k.

$m \leftarrow \lfloor \sqrt{nk} \rfloor$
$R \leftarrow \lfloor \frac{1}{2} \log(n/k) - 2 \rfloor$ ▷ Note that $R \geq 3$ because $k \leq \frac{n}{2^{10}}$
$T_0 \leftarrow [n]$
for $r = 1, \ldots, R$ **do**
  $T_r \leftarrow \emptyset$
  For each $a \in T_{r-1}$, $T_r \leftarrow T_r \cup \{a\}$ with probability $\frac{1}{2}$ ▷ We have $S_r = S \cap T_r$
end for

Algorithm 6 Encoder ENC_k.

procedure $\mathrm{ENC}_k(S)$
  $M \leftarrow \mathsf{Alice}_k(\mathbf{1}_S)$
  $A \leftarrow \emptyset$
  for $r = 1, \ldots, R$ **do**
    $A_r \leftarrow \mathsf{Bob}_k(M, \mathbf{1}_{S \setminus (S \cap T_{r-1})})$
    if $A_r \subseteq S \cap T_{r-1}$ **then** ▷ i.e. if A_r is valid
      $b_r \leftarrow 1$ ▷ b is a binary string of length R, indicating if Bob_k succeeds in round r
      $A \leftarrow A \cup (A_r \cap (T_{r-1} \setminus T_r))$
    else
      $b_r \leftarrow 0$
    end if
  end for
  return $(M,\, S \setminus A,\, b)$
end procedure

Algorithm 7 Decoder DEC_k.

procedure $\text{DEC}_k(M,\ B,\ b)$
  $A \leftarrow \emptyset$
  $C_0 \leftarrow \emptyset$
  for $r = 1, \dots, R$ **do**
    $C_r \leftarrow C_{r-1}$
    if $b_r = 1$ **then**
      $A_r \leftarrow \mathsf{Bob}_k(M, \mathbf{1}_{C_{r-1}})$ ▷ Invariant: $C_r = S \backslash (S \cap T_r)$
      $A \leftarrow A \cup (A_r \cap (T_{r-1} \backslash T_r))$
      $C_r \leftarrow C_r \cup (A_r \cap (T_{r-1} \backslash T_r))$
    end if
    $C_r \leftarrow C_r \cup (B \cap (T_{r-1} \backslash T_r))$
  end for
  return $B \cup A$
end procedure

3.2.2 Analysis

Theorem 3. $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k^\subset) = \Omega((1-\delta)k \log^2 \frac{n}{k})$, *given that* $1 \le k \le \frac{n}{2^{10}}$ *and* $0 < \delta \le 1 - \frac{50 \log n}{k \log^2(n/k)}$.

Proof. Let $S_r = S \cap T_r$. Let SUCC denote the event that $|S \cap T_R| = |S_R| \ge k$. Note that $\mathbb{E}\,|S_R| = \frac{1}{2^R} m = 4k$. By the Chernoff bound, $\mathbb{P}(\mathsf{SUCC}) \ge \frac{1}{2}$. In the following, we argue conditioned on SUCC. Namely, in each round r, there are at least k items in S_r.

Similar to Lemma 3, we can prove the protocol $(\mathsf{ENC}_k, \mathsf{DEC}_k)$ always succeeds. By Lemma 2, we have $\mathsf{s} \ge \log\binom{n}{m} - \mathsf{s}' - 2$, where $\mathsf{s}' = \log n + R + \mathbb{E}\log\binom{n}{|B|}$. The size of B is $|B| = |S| - \sum_{r=1}^R (b_r \cdot |A_r \cap (S_{r-1} \backslash S_r)|)$. The randomness used by $\mathcal{P}$ is independent from $S \backslash S_{r-1}$ for every $r \in [R]$. Therefore, $\mathbb{E}\,b_r \ge 1 - \delta$, and b_r is independent from $|A_r \cap (S_{r-1} \backslash S_r)|$. We have $\mathbb{E}\,|A_r \cap (S_{r-1} \backslash S_r)| = \frac{k}{2}$, and thus $\mathbb{E}(|S| - |B|) \ge \frac{(1-\delta)kR}{2}$. By Lemma 4, $\log\binom{n}{m} - \mathbb{E}\log\binom{n}{|B|} \ge \frac{(1-\delta)kR}{2} \cdot \log(\frac{n}{m} - 1) \ge \frac{(1-\delta)kR}{5}\log(\frac{n}{k})$. Moreover, $R \le \log n$ and $\log n \le \frac{(1-\delta)kR}{12} \log \frac{n}{k}$. Thus we have $\mathsf{s} = \Omega((1-\delta)kR \log \frac{n}{k}) = \Omega((1-\delta)k \log^2 \frac{n}{k})$. □

4 Lower bounds proofs via augmented indexing

Here we show another route to proving $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k^\subset) = \Omega(\min\{n, t\log^2(n/t)\}$ via reduction from augmented indexing. We again separately prove lower bounds for $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^\subset)$ and $\mathbf{R}_{\frac{1}{5}}^{\to,pub}(\mathbf{UR}_k^\subset)$. Both proofs make use of the following standard lemma. The proof can be found in the appendix (see Section A.2).

Lemma 7. *For any integers $u \ge 1$ and $1 \le m \le u/(4e)$, there exists a collection $\mathcal{S}_{u,m} \subset \binom{[u]}{m}$ with $\log|\mathcal{S}_{u,m}| = \Theta(m \log(u/m))$ such that for all $S \ne S' \in \mathcal{S}_{u,m}$, $|S \cap S'| < m/2$.*

Both our lower bounds in Sections 4.1 and 4.2 reduce from augmented indexing (henceforth **AugIndex**) to either $\mathbf{UR}^\subset$ with low failure probability, or $\mathbf{UR}_k^\subset$ with constant failure probability, in the public coin one-way model of communication. We remind the reader of the setup for the $\mathbf{AugIndex}_N$ problem. There are two players, Charlie and Diane. Charlie receives $z \in \{0,1\}^N$ and

Diane receives $j^* \in [N]$ together with $z_{j^*+1}, \ldots, z_N$. Charlie must send a single message to Diane such that Diane can then output z_{j^*}. The following theorem is known.

Theorem 4. *[MNSW98]* $\mathbf{R}_{1/3}^{\to,pub}(\mathbf{AugIndex}_N) = \Theta(N)$.

We show that if there is an s-bit communication protocol $\mathcal{P}$ for $\mathbf{UR}^{\subset}$ on n-bit vectors with failure probability δ (or for $\mathbf{UR}_k$ with constant failure probability), that implies the existence of an s-bit protocol $\mathcal{P}'$ for $\mathbf{AugIndex}_N$ for some $N = \Theta(\log\frac{1}{\delta}\log^2\frac{n}{\log\frac{1}{\delta}})$ (or $N = \Theta(k\log^2(n/k))$ for $\mathbf{UR}_k$). The lower bound on s then follows from Theorem 4.

4.1 Communication Lower Bound for $\mathbf{UR}^{\subset}$

Set $t = \log\frac{1}{\delta}$. In this section we assume $t < n/(4e)$ and show $\mathbf{R}_{\delta}^{\to,pub}(\mathbf{UR}^{\subset}) = \Omega(t\log^2(n/t))$. This implies a lower bound of $\Omega(\min\{n, t\log^2(n/t)\})$ for all $\delta > 0$ bounded away from 1.

As mentioned, we assume we have an s-bit protocol $\mathcal{P}$ for $\mathbf{UR}^{\subset}$ with failure probability δ, with players Alice and Bob.We use $\mathcal{P}$ to give an s-bit protocol $\mathcal{P}'$ for $\mathbf{AugIndex}_N$, which has players Charlie and Diane, for $N = \Theta(t\log^2(n/t))$.

The protocol $\mathcal{P}'$ operates as follows. Without loss of generality we may assume that, using the notation of Lemma 7, $|\mathcal{S}_{u,m}|$ is a power of 2 for u, m as in the lemma statement. This is accomplished by simply rounding $|\mathcal{S}_{u,m}|$ down to the nearest power of 2 by removing elements arbitrarily. Also, define $L = c\log(n/t)$ for some sufficiently small constant $c \in (0,1)$ to be determined later. Now, Charlie partitions the bits of his input $z \in \{0,1\}^N$ into L consecutive sequences of bits such that the ith chunk of bits for each $i \in [L]$ can be viewed as specifying an element $S_i \in \mathcal{S}_{u_i,m}$ for $u_i = \frac{n}{100^i \cdot L}$ and $m = ct$. Lemma 7 gives $\log|\mathcal{S}_{u_i,m}| = \Theta(m\log(u_i/m))$, which is $\Theta(t\log(n/t))$ for $c < 1/14$. Thus $N = \Theta(L \cdot t\log(n/t)) = \Theta(t\log^2(n/t))$. Given these sets $S_1, \ldots, S_L$, we now discuss how Charlie generates a vector $x \in \{0,1\}^n$. Charlie then simulates Alice on x to generate the message Alice would have send to Bob in protocol $\mathcal{P}$, then sends that same message to Diane.

To generate $x \in \{0,1\}^n$, assume Charlie and Diane have sampled a bijection from

$$A = \bigcup_{i=1}^{L}(\{i\} \times [u_i] \times [100^i]) \tag{10}$$

to $[n]$ uniformly at random. We denote this bijection by π. This is possible since $|A| = n$. Then Charlie defines x to be the indicator vector $\mathbf{1}_{\pi(S)}$, where

$$S = \bigcup_{i=1}^{L}(\{i\} \times S_i \times [100^i]),$$

then sends a message M to Diane, equal to Alice's message with input $\mathbf{1}_{\pi(S)}$. This completes the description of Charlie's behavior in the protocol $\mathcal{P}'$.

We describe how Diane uses M to solve $\mathbf{AugIndex}_N$. Diane's input $j^* \in [N]$ lies in some chunk $i^* \in [L]$. We now show how Diane can use $\mathcal{P}$ to recover S_{i^*} with probability 2/3 (and thus in particular recover z_{j^*}). Since Diane knows z_j for $j > j^*$, she knows S_i for $i > i^*$. She can then execute the following algorithm.

Algorithm 8 Behavior of Diane in $\mathcal{P}'$ for $\mathbf{UR}^{\subset}$.

procedure $\mathsf{Diane}(M)$
  $T \leftarrow \bigcup_{i=i^*+1}^{L}(\{i\} \times S_i \times [100^i])$
  $T_{i^*} \leftarrow \emptyset$
  while $|T_{i^*}| < \frac{m}{2}$ **do**
    $(i, a, r) \leftarrow \pi^{-1}(\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}))$
    $T \leftarrow T \cup ((i, a) \times [100^i])$
    if $i = i^*$ **then**
      $T_{i^*} \leftarrow T_{i^*} \cup \{a\}$
    end if
  end while
  if there exists $S \in \mathcal{S}_{u_{i^*},m}$ with $T_{i^*} \subset S$ **then**
    return the unique such S
  else
    return Fail
  end if
end procedure

In Algorithm 8 Diane is building up a subset T_{i^*} of S_{i^*}. Once $|T_{i^*}| \geq |S_{i^*}|/2 = m/2$, Diane can uniquely recover S_{i^*} by the limited intersection property of $\mathcal{S}_{u_i,m}$ guaranteed by Lemma 7. Until then, she uses $\mathcal{P}$ to recover elements of $S \backslash T$, which, as we now show, are chosen uniformly at random from $S \setminus T$.

Claim 4. *For every protocol for Alice and Bob that uses shared randomness with Bob's behaviour given by* $\mathsf{Bob}(\cdot)$*, for every choice of shared random string R of Alice and Bob, for every $S, T \subset S$, the following conditions hold. If π is a uniformly random permutation, the success or failure of* $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ *is determined by $\{\pi(j)\}_{j \in T}$ and the image $\pi(S \setminus T)$ of $S \setminus T$ under π. Conditioned on a choice of R, $\{\pi(j)\}_{j \in T}$ and $\pi(S \setminus T)$ such that* $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ *succeeds, one has that* $\pi^{-1}(\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}))$ *is a uniformly random element of $S \setminus T$.*

Proof. The first claim follows by noting that the message M that Alice sends to Bob is solely a function of R and $\pi(S)$. The behaviour of Bob is determined by M and $\pi(T)$ (and the latter is determined by $\{\pi(j)\}_{j \in T}$).

Now condition on the values of R, $\{\pi(j)\}_{j \in T}$ and $\pi(S \setminus T)$ such that $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ succeeds, and let $j^* \in [n]$ denote the output. Note that by our conditioning j^* is a fixed quantity. The only randomness left is the exact mapping of $S \setminus T$ to $\pi(S \setminus T)$. This mapping is independent of $\{\pi(j)\}_{j \in T}$ and $\pi(S \setminus T)$ and uniformly random, so $\pi^{-1}(j^*)$ is a uniformly random element of $S \setminus T$, as required. ◻

Fix any protocol $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ (not necessarily the one that Charlie and Diane use; see analysis of the idealized process $\widetilde{\mathcal{P}}$ below). Now fix T together with values of R, $\{\pi(j)\}_{j \in T}$ and $\pi(S \setminus T)$ such that $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ succeeds.

Elements in $S_j, j < i^*$, are unlikely to be recovered. Given Claim 4, since the elements of S_j appear with frequency 100^j in $S \backslash T$, they are less likely to be returned by $\pi^{-1}(\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)}))$

when j is small. More precisely, as long as $|S_{i^*} \cap T_{i^*}| \geq m/2$, for any $1 \leq j < i^*$

$$\begin{aligned}\mathbb{P}(i = j | (R, \{\pi(j)\}_{j \in T}, \pi(S \setminus T)) \text{ s.t. } \widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)}) \text{ succeeds}) &\leq \frac{m \cdot 100^j}{\frac{m}{2} \cdot 100^{i^*}} \\ &\leq 2 \cdot 100^{-(i^*-j)} \\ &\leq 50^{-(i^*-j)}.\end{aligned} \tag{11}$$

Here again the probability is over the choice of $\pi|_{S \setminus T} : (S \setminus T) \to \pi(S \setminus T)$ (recall that we condition on the image $\pi(S \setminus T)$ under π, but not on the actual mapping).

We now define the set $\mathcal{T}$ of **typical intermediate sets**, which plays a crucial role in our analysis. Let Q_i for $i \in [L]$ denote $\{i\} \times S_i \times [100^i]$. Let $\mathcal{T}$ be the collection of all $T \subset S$ such that (1) $Q_i \subset T$ for all $i > i^*$, and (2) for each $i < i^*$, $|T \cap Q_i| \leq 100^i \cdot m/4^{i^*-i}$. The following claim will be useful:

Claim 5. *For the set $\mathcal{T}$ defined above one has $|\mathcal{T}| = 2^{O(m)}$.*

Proof.

$$\begin{aligned}|\mathcal{T}| &\leq 2^m \cdot \prod_{i=1}^{i^*-1} \left(\sum_{r=0}^{\frac{m}{4^{i^*-i}}} \binom{m}{r} \right) \quad \text{(the } 2^m \text{ term comes from } S_{i^*}) \\ &\leq 2^m \cdot \prod_{i=1}^{i^*-1} \binom{m + \frac{m}{4^{i^*-i}}}{\frac{m}{2^{i^*-i}}} \\ &\leq 2^m \cdot \prod_{i=1}^{i^*-1} (2e \cdot 4^{i^*-i})^{\frac{m}{4^{i^*-i}}} \quad \text{(using } \binom{n}{k} \leq (en/k)^k) \\ &\leq 2^{O(m)} \cdot 2^{m \cdot O(\sum_{j=1}^{\infty} j 4^{-j})} \\ &\leq 2^{O(m)}\end{aligned}$$

□

We will show that for most choices of π and shared random string R Algorithm 8 **(a)** never leaves the set $\mathcal{T}$ and **(b)** successfully terminates. Note that Algorithm 8 is a random process whose sample space is the product of the set of all possible permutations π and shared random strings R. As before, we denote this process by $\mathcal{P}'$. It is useful for analysis purposes to define another process $\widetilde{\mathcal{P}}$, which is an idealized version of $\mathcal{P}'$. In this process instead of running $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ Alice runs $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$, which is guaranteed to output an element of $\pi(S \setminus T)$ for every choice of $T \subset S$, shared random string R, $\{\pi(j)\}_{j \in T}$, and $\pi(S \setminus T)$. The proof proceeds in three steps.

Step 1: proving that $\widetilde{\mathcal{P}}$ succeeds in recovering T_{i^*} and never leaves $\mathcal{T}$ with high probability. Choose π uniformly at random. By (11), as long as $|S_{i^*} \cap T_{i^*}| \geq m/2$, the expected number of items recovered by $\widetilde{\mathsf{Bob}}$ from S_i for $i < i^*$ in the first m iterations is at most $m/50^{i^*-i}$. Thus the probability of recovering more than $m/4^{i^*-i}$ items from S_i is at most $(1/12)^{i^*-i}$ by Markov's inequality. Note that the probability is over the choice of π only, as $\widetilde{\mathsf{Bob}}$ is assumed to succeed with probability 1 by definition of $\widetilde{\mathcal{P}}$. Thus

$$\mathbb{P}(\widetilde{\mathcal{P}} \text{ leaves } \mathcal{T}) \leq \sum_{i=1}^{i^*-1} (1/12)^{i^*-i} < 1/10.$$

In particular this means that with probability at least $1-1/10$ at most $\sum_{i<i^*} m/4^{i^*-i} < m/2$ items from $\bigcup_{i<i^*} S_i$ are recovered in the first m (or fewer, if the algorithm terminates earlier) iterations. This also implies that with probability at least $1-1/10$ if the algorithm proceeds for the entire m iterations, it recovers at least $m/2$ elements of T_{i^*} and hence terminates. We thus get that $\widetilde{\mathcal{P}}$ succeeds at least with probability $1-1/10$.

Step 2: coupling $\widetilde{\mathcal{P}}$ to $\mathcal{P}'$ on most of the probability space. For every $T \subset S$ and every π let $\mathcal{E}_T(\pi)$ be the probabilistic event (over the choice of Bob's random string R) that $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ succeeds in returning an element in $\pi(S \backslash T)$. Note that $\mathcal{E}_T(\pi)$ is a subset of the probability space of shared random strings R, and depends on π. We let

$$\mathcal{E}_{\mathcal{T}}(\pi) := \wedge_{T \in \mathcal{T}} \mathcal{E}_T(\pi)$$

to simplify notation. Using Claim 5 and the union bound we have for every π

$$\mathbb{P}_R(\neg(\mathcal{E}_{\mathcal{T}}(\pi))) \le \delta \cdot |\mathcal{T}| \le 1/20$$

as long as for $m = c\log(1/\delta)$ for c a sufficiently small constant.

Now recall that $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ is an idealized protocol, which is guaranteed to output an element of $\pi(S \setminus T)$ for every choice of $T \subset S$, shared random string R, $\{\pi(j)\}_{j \in T}$, and $\pi(S \setminus T)$. We have just shown that for every π the event $\mathcal{E}_{\mathcal{T}}(\pi)$ occurs with probability at least $1-1/20$ over the choice of R. Now define $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ as equal to $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ for all $T \in \mathcal{T}$ (the typical set of intermediate sets) and (π, R) such that $R \in \mathcal{E}_{\mathcal{T}}(\pi)$, and extend $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ to return an arbitrary element of $\pi(S \setminus T)$ for remaining tuples $(T, R, \pi(T), \pi(S \setminus T))$. Note that $\widetilde{\mathsf{Bob}}$ defined in this way is a deterministic function once T, R, $\pi(T)$ and $\pi(S \setminus T)$ are fixed.

Note that with probability at least $1-1/20$ over the choice of π and R one has $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}) = \widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ for all $T \in \mathcal{T}$, as required.

Step 3: arguing that $\mathcal{P}'$ succeeds with high probability. Choose (π, R) uniformly at random. By **Step 2** we have that with probability at least $1-1/20$ over this choice $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}) = \widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ for all $T \in \mathcal{T}$. At the same time we have by **Step 1** that with probability at least $1-1/10$ over the choice of π the idealized process $\widetilde{\mathcal{P}}$ succeeds in recovering T_{i^*} and never leaves $\mathcal{T}$. Putting the two bounds together, we get that $\mathcal{P}'$ succeeds with probability at least $1-1/20-1/10 > 2/3$, showing the following theorem.

Theorem 5. *For any $0 < \delta < 1/2$ and integer $n \ge 1$ with $\log\frac{1}{\delta} < n/(4e)$, $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^{\subset}) \ge \mathbf{R}_{1/3}^{\to,pub}(\mathbf{AugIndex}_N)$ for $N = \Theta(\log\frac{1}{\delta}\log^2\frac{n}{\log\frac{1}{\delta}})$.*

Corollary 2. *For any $0 < \delta < 1/2$ and integer $n \ge 1$, $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}^{\subset}) = \Omega(\min\{n, \log\frac{1}{\delta}\log^2\frac{n}{\log\frac{1}{\delta}}\})$.*

4.2 Communication Lower Bound for $\mathbf{UR}_k^{\subset}$

The idea for lower bounding $\mathbf{R}_{\frac{1}{5}}^{\to,pub}(\mathbf{UR}_k^{\subset})$ is as in Section 4.1, but slightly simpler. That is because for the protocol $\mathcal{P}'$ for $\mathbf{AugIndex}_N$, Diane will not make adaptive queries to Bob in the protocol $\mathcal{P}$ for $\mathbf{UR}_k^{\subset}$. Rather, she will only make one query using Bob and will be able to decide $\mathbf{AugIndex}_N$ with good probability from that single query. We make use of the following lemma from [JST11], whose proof is similar to our analysis in Section 4.1.

Lemma 8. *[JST11] Any public coin protocol for* $\mathbf{UR}^{\subset}$ *can be turned into one that outputs every index* $i \in [n]$ *with* $x_i \neq y_i$ *with the same probability. The number of bits sent, failure probability, and number of rounds do not change. Similarly, any* $\mathbf{UR}_k^{\subset}$ *protocol can be turned into one in which all subsets of* $[n]$ *of size* $\min\{k, \|x-y\|_0\}$ *on which* x, y *differ are equally likely to be output.*

Henceforth we assume $\mathcal{P}$ outputs random differing indices, which is without loss of generality by Lemma 8.

Again Charlie receives $z \in \{0,1\}^N$ and Diane receives j^* and $z_{j^*+1}, \ldots, z_N$ and they want to solve $\mathbf{AugIndex}_N$. Charlie views his input as consisting of L blocks for $L = c\log(n/k)$ for a sufficiently small constant $c \in (0,1)$, and the ith block for $i \in [L]$ specifies a set $S_i \in \mathcal{S}_{u_i,m}$ for $m = ck$ and $u_i = n/(100^i L)$. As before, for c sufficiently small we have $N = \Theta(L \cdot k\log(n/k)) = \Theta(k\log^2(n/k))$. The bijection A and set S are defined exactly as in Section 4.1, and Charlie simulates Alice to send the message M to Diane that Alice would have sent to Bob on input $\mathbf{1}_S$. Again, Diane knows S_i for $i > i^*$, where j^* lies in the i^*th block of bits. Diane's algorithm to produce her output is then described in Algorithm 9.

Algorithm 9 Behavior of Diane in $\mathcal{P}'$ for $\mathbf{UR}_k^{\subset}$.

procedure $\mathsf{Diane}(M)$
  $T \leftarrow \bigcup_{i=i^*+1}^{L}(\{i\} \times S_i \times [100^i])$
  $T_{i^*} \leftarrow \emptyset$
  $B \leftarrow \mathsf{Bob}(M, \mathbf{1}_T)$
  for $(i, a, r) \in B$ **do**
    if $i = i^*$ and $a \notin T$ **then**
      $T_{i^*} \leftarrow T_{i^*} \cup \{a\}$
    end if
  end for
  if $|T_{i^*}| < \frac{m}{2}$ **then**
    return Fail
  else
    return the unique $S \in \mathcal{S}_{u_{i^*},m}$ with $T_{i^*} \subset S$
  end if
end procedure

Recall Bob, when he succeeds, returns $\min\{k, |S\backslash T|\} = k$ uniformly random elements from $S\backslash T$. Meanwhile, S_{i^*} only has $m = ck$ elements for some small constant c. As in Section 4.1, almost all of the support of $S\backslash T$ comes from items in block i^*, and hence we expect almost all our k samples to come from (and be uniform in) items corresponding to elements of S_{i^*}.

We now provide a formal analysis. Henceforth we condition on Bob succeeding, which happens with probability 4/5. The number of elements in $S\backslash T$ corresponding to an element of S_{i^*} is $100^{i^*}m$, whereas the number of elements corresponding to an element of S_i for $i < i^*$ is

$$m \cdot \sum_{i=1}^{i^*-1} 100^i = \frac{m}{99} \cdot (100^{i^*} - 1) < \frac{m}{99} \cdot 100^{i^*}$$

Thus, we expect at most $k/99$ elements in B to correspond to elements in S_i for $i \neq i^*$, and the probability that we have at least $k/9$ such elements in B is less than 1/10 by Markov's inequality.

We henceforth condition on having less than $k/9$ such elements in B. Now we know B contains at least $8k/9$ elements corresponding to S_{i^*}, chosen uniformly from $S_{i^*} \times [100^i]$. For any given element $a \in S_{i^*}$, the probability that none of the elements in B from S_{i^*} correspond to a is $(1-1/m)^{\frac{8}{9}k} \leq e^{-(8/9)k/m} < 1/30$ for c sufficiently small (where $m = ck$). Thus the expected number of $a \in S_{i^*}$ not covered by B is less than $m/30$. Thus the probability that fewer than $m/2$ elements are covered by B is a most $1/15$ by Markov's inequality (and otherwise, Diane succeeds). Thus, the probability that Diane succeeds is at least $4/5 \cdot 9/10 \cdot 14/15 > 2/3$. We have thus shown the following theorem.

Theorem 6. *For any integers* $1 \leq k \leq n$, $\mathbf{R}^{\rightarrow,pub}_{\frac{1}{5}}(\mathbf{UR}^{\subset}_k) \geq \mathbf{R}^{\rightarrow,pub}_{\frac{1}{3}}(\mathbf{AugIndex}_N)$ *for* $N = \Theta(k\log^2(n/k))$.

Corollary 3. *For any integers* $1 \leq k \leq n$, $\mathbf{R}^{\rightarrow,pub}_{\frac{1}{5}}(\mathbf{UR}^{\subset}_k) = \Omega(\min\{n, k\log^2(n/k)\})$.

Remark 1. One may wish to understand $\mathbf{R}^{\rightarrow,pub}_{\delta}(\mathbf{UR}^{\subset}_k)$ for δ near 1 (or at least, larger than $1/2$). Such a lower bound is given in Theorem 3. The proof given above as written would yield no lower bound in this regime for δ since **AugIndex** is in fact easy when the failure probability is allowed to be least $1/2$ (Charlie can send no message at all, and Diane can simply guess z_{j^*} via a coin flip). One can however get a handle on $\mathbf{R}^{\rightarrow,pub}_{\delta}(\mathbf{UR}^{\subset}_k)$ by instead directly reducing from the following variant of augmented indexing: Charlie receives $D \in \mathcal{S}_{u_1,m} \times \cdots \times \mathcal{S}_{u_L,m}$ and Diane receives $j^* \in [L]$ and $D_{j^*+1}, \ldots, D_L$ and must output D_{j^*}, where the u_i are as above. One can show that unless Charlie sends almost his entire input, Diane cannot have success probability significantly better than random guessing (which has success probability $O(\max_{i\in L} 1/|\mathcal{S}_{u_i,m}|)$). The proof is nearly identical to the analysis of augmented indexing over large domains [EJS10, JW13]. Indeed, the problem is even almost identical, except that here we consider Charlie receiving a vector whose entries come from different alphabet sizes (since the $|\mathcal{S}_{u_i,m}|$ are different), whereas in [EJS10, JW13] all the entries come from the same alphabet.

Acknowledgments

Initially the authors were focused on proving optimal lower bounds for samplers, but we thank Vasileios Nakos for pointing out that our $\mathbf{UR}^{\subset}$ lower bound immediately implies a tight lower bound for finding a duplicate in data streams as well. Also, initially our proof of Lemma 1 incurred an additive 1 in the numerator of the right hand side of (2). This is clearly suboptimal for small $I(X;Y)$ (for example, consider $I(X;Y) = 0$, in which case the right hand side should be δ and not $1/\log(1/\delta))$). We thank T.S. Jayram for pointing out that a slight modification of our proof could actually replace the additive 1 with the binary entropy function (and also for showing us a different proof of this lemma, which resembles the standard proof of Fano's inequality).

References

[AGM12a] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–467, 2012.

[AGM12b] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31^{st} ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 5–14, 2012.

[AGM13] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *Proceedings of the 16^{th} International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–10, 2013.

[AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the 28^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1723–1742, 2017.

[AKLY16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the 27^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1345–1364, 2016.

[AKO11] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *Proceedings of the 52^{nd} Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 363–372, 2011.

[BHNT15] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the 47^{th} Annual ACM on Symposium on Theory of Computing (STOC)*, pages 173–182, 2015.

[BS15] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Proceedings of the 23^{rd} Annual European Symposium on Algorithms (ESA)*, pages 263–274, 2015.

[CCE$^+$16] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the 27^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1344, 2016.

[CCHM15] Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the 26^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1234–1251, 2015.

[CF14] Graham Cormode and Donatella Firmani. A unifying framework for ℓ_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014. Preliminary version in ALENEX 2013.

[CK04] Don Coppersmith and Ravi Kumar. An improved data stream algorithm for frequency moments. In *Proceedings of the 15^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 151–156, 2004.

[CMR05] Graham Cormode, S. Muthukrishnan, and Irina Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 25–36, 2005.

[DM16] Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. In *Proceedings of the 31st Conference on Computational Complexity (CCC)*, pages 3:1–3:51, 2016.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[EHW16] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. Brief announcement: Applications of uniform sampling: Densest subgraph and beyond. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 397–399, 2016.

[EIRS91] Jack Edmonds, Russell Impagliazzo, Steven Rudich, and Jiri Sgall. Communication complexity towards lower bounds on circuit depth. In *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 249–257, 1991.

[EJS10] Funda Ergün, Hossein Jowhari, and Mert Sağlam. Periodicity in streams. In *Proceedings of the 14th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 545–559, 2010.

[FIS08] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. *Int. J. Comput. Geometry Appl.*, 18(1/2):3–28, 2008. Preliminary version in SOCG 2005.

[FT16] Martin Farach-Colton and Meng-Tsung Tsai. Tight approximations of degeneracy in large graphs. In *Proceedings of the 12th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 429–440, 2016.

[GKKT15] David Gibb, Bruce M. Kapron, Valerie King, and Nolan Thorn. Dynamic graph connectivity with improved worst case update time and sublinear space. *CoRR*, abs/1509.06464, 2015.

[GMT15] Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS)*, pages 241–247, 2015.

[GMWW14] Dmitry Gavinsky, Or Meir, Omri Weinstein, and Avi Wigderson. Toward better formula lower bounds: an information complexity approach to the KRW composition conjecture. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 213–222, 2014.

[GR09] Parikshit Gopalan and Jaikumar Radhakrishnan. Finding duplicates in a data stream. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 402–411, 2009.

[HPP+15] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *Proceedings of the 34th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 91–100, 2015.

[HW90] Johan Håstad and Avi Wigderson. Composition of the universal relation. In *Proceedings of a DIMACS Workshop on Advances In Computational Complexity Theory*, pages 119–134, 1990.

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 49–58. ACM, 2011.

[JW13] T. S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Trans. Algorithms*, 9(3):26:1–26:17, 2013.

[KKM13] Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1131–1142, 2013.

[KLM+14] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 561–570, 2014.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2010.

[Kon15] Christian Konrad. Maximum matching in turnstile streams. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 840–852, 2015.

[KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3-4):191–204, 1995.

[KW90] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990.

[McG14] Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[MNSW98] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.

[MTVV15] Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 472–482, 2015.

[Mut05] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[MW10] Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error l_p-sampling with applications. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1143–1160, 2010.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NPW17] Jelani Nelson, Jakub Pachocki, and Zhengyu Wang. Optimal lower bounds for universal relation, samplers, and finding duplicates. *CoRR*, abs/1703.08139, March 2017.

[PRS16] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast distributed algorithms for connectivity and MST in large graphs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 429–438, 2016.

[Tar07] Jun Tarui. Finding a duplicate and a missing item in a stream. In *Proceedings of the 4th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 128–135, 2007.

[TZ97] Gábor Tardos and Uri Zwick. The communication complexity of the universal relation. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity (CCC)*, pages 247–259, 1997.

[Wan15] Zhengyu Wang. An improved randomized data structure for dynamic graph connectivity. *CoRR*, abs/1510.04590, 2015.

A Appendix

A.1 A tight upper bound for $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k)$

In [JST11, Proposition 1] it is shown that $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k) = O(\min\{n, t\log^2 n\})$ for $t = \max\{k, \log(1/\delta)\}$. Here we show that a minor modification of their protocol in fact shows the correct complexity $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k) = O(\min\{n, t\log^2(n/t)\})$, which given our new lower bound, is optimal up to a constant factor for the full range of n, k, δ as long as δ is bounded away from 1.

Recall Alice and Bob receive $x, y \in \{0,1\}^n$, respectively, and share a public random string. Alice must send a single message M to Bob, from which Bob must recover $\min\{k, \|x-y\|_0\}$ indices $i \in [n]$ for which $x_i \neq y_i$. Bob is allowed to fail with probability δ. The fact that $\mathbf{R}_\delta^{\to,pub}(\mathbf{UR}_k) \le n$ is obvious: Alice can simply send the message $M = x$, and Bob can then succeed with failure probability 0. We thus now show $\mathbf{R}_{e^{-ck}}^{\to,pub}(\mathbf{UR}_k) \le k\log^2(n/k)$ for some constant $c > 0$, which completes the proof of the upper bound. We assume $k \le n/2$ (otherwise, Alice sends x explicitly).

As mentioned, the protocol we describe is nearly identical to one in [JST11] (see also [CF14]). We will describe the new protocol here, then point out the two minor modifications that improve the $O(k\log^2 n)$ bound to $O(k\log^2(n/k))$ in Remark 2. We first need the following lemma.

Lemma 9. *Let $\mathbb{F}_q$ be a finite field and $n > 1$ an integer. Then for any $1 \le k \le \frac{n}{2}$, there exists $\Pi_k \in \mathbb{F}_q^{m\times n}$ for $m = O(k\log_q(qn/k))$ s.t. for any $w \neq w' \in \mathbb{F}_q^n$ with $\|w\|_0, \|w'\|_0 \le k$, $\Pi_k w \neq \Pi_k w'$.*

Proof. The proof is via the probabilistic method. $\Pi_k w = \Pi_k w'$ iff $\Pi_k(w - w') = 0$. Note $v = w - w'$ has $\|v\|_0 \le 2k$. Thus it suffices to show that such a Π_k exists with no $(2k)$-sparse vector in its kernel. The number of vectors $v \in \mathbb{F}_q^n$ with $\|v_0\| \le 2k$ is at most $\binom{n}{2k}\cdot q^{2k}$. For any fixed v, $\mathbb{P}(\Pi_k v = 0) = q^{-m}$. Thus

$$\mathbb{P}(\exists v, \|v\|_0 \le 2k : \Pi_k v = 0) \le \binom{n}{2k}\cdot q^{2k}\cdot q^{-m}$$

by a union bound. The above is strictly less than 1 for $m > 2k + \log_q\binom{n}{2k}$, yielding the claim. $\square$

Corollary 4. *Let $\mathbb{F}_q$ be a finite field and $n > 1$ an integer. Then for any $1 \le k \le \frac{n}{2}$, there exists $\Pi_k \in \mathbb{F}_q^{m\times n}$ for $m = O(k\log_q(qn/k))$ together with an algorithm $\mathcal{R}$ such that for any $w \in \mathbb{F}_q^n$ with $\|w\|_0 \le k$, $\mathcal{R}(\Pi_k w) = w$.*

Proof. Given Lemma 9, a simple such $\mathcal{R}$ is as follows. Given some $y = \Pi_k w^*$ with $\|w^*\|_0 \le k$, $\mathcal{R}$ loops over all w in $\mathbb{F}_q^n$ with $\|w\|_0 \le k$ and outputs the first one it finds for which $\Pi_k w = y$. $\square$

The protocol for $\mathbf{UR}_k$ is now as follows. Alice and Bob use public randomness to pick commonly known random functions $h_0, \ldots, h_L : [n] \to \{0,1\}$ for $L = \lfloor \log_2(n/k)\rfloor$, such that for any $i \in [n]$ and for any j, $\mathbb{P}(h_j(i) = 1) = 2^{-j}$. They also agree on a matrix Π_{16k} and $\mathcal{R}$ as described in Corollary 4 for a sufficiently large constant $C > 0$ to be determined later, with $q = 3$. Thus Π_{16k} has $m = O(k\log(n/k))$ rows. Alice then computes $v_j = \Pi_{16k} x|_{h_j^{-1}(1)}$ for $j = 0, \ldots, L$ where $v_j \in \mathbb{F}_q^m$, and her message to Bob is $M = (v_0, \ldots, v_L)$. For $S \subseteq [n]$ and x an n-dimensional vector, $x|_S$ denotes the n-dimensional vector with $(x|_S)_i = x_i$ for $i \in S$, and $(x|_S)_i = 0$ for $i \notin S$. Note Alice's message M is $O(k\log^2(n/k))$ bits, as desired. Bob then executes the following algorithm and outputs the returned values.

Algorithm 10 Bob's algorithm in the $\mathbf{UR}_k$ protocol.

procedure Bob$(v_0, \ldots, v_L)$
    for $j = L, L-1, \ldots, 0$ **do**
        $v_j \leftarrow v_j - \Pi_{16k} y|_{h_j^{-1}(1)}$
        $w_j \leftarrow \mathcal{R}(v_j)$
        if $\|w_j\|_0 \ge k$ or $j = 0$ **then**
            return an arbitrary $\min\{k, \|w_j\|_0\}$ elements from *support*(w_j)
        end if
    end for
end procedure

The correctness analysis is then as follows, which is nearly the same as the ℓ_0-sampler of [JST11]. If Alice's input is x and Bob's is y, let $a = x - y \in \{-1, 0, 1\}^n$, so that a can be viewed as an element of $\mathbb{F}_3^n$. Also let $a_j = a|_{h_j^{-1}(1)}$. Then $\mathbb{E}\,\|v_j\|_0 = \|a\|_0 \cdot 2^{-j}$, and since $0 \le \|a\|_0 \le n$, there either (1) exists a unique $0 \le j^* \le L$ such that $2k \le \mathbb{E}\,\|a_j\|_0 \cdot 2^{-j^*} < 4k$, or (2) $\|a\|_0 < 2k$ (in which case we define $j^* = 0$). Let $\mathcal{E}$ be the event that $\|a_j\|_0 \le 16k$ simultaneously for all $j \ge j^*$. Let $\mathcal{F}$ be the event that *either* we are in case (2), or we are in case (1) and $\|a_{j^*}\|_0 \ge k$ holds. Note that conditioned on $\mathcal{E}, \mathcal{F}$ both occurring, Bob succeeds by Corollary 4.

We now just need to show $\mathbb{P}(\neg\mathcal{E} \wedge \neg\mathcal{F}) < e^{-\Omega(k)}$. We use the union bound. First, consider $\mathcal{F}$. If $j^* = 0$, then $\mathbb{P}(\neg\mathcal{F}) = 0$. If $j^* \neq 0$, then $\mathbb{P}(\neg\mathcal{F}) \le \mathbb{P}(\|a_{j^*}\|_0 < \frac{1}{2} \cdot \mathbb{E}\,\|a_{j^*}\|_0)$, which is $e^{-\Omega(k)}$ by the Chernoff bound since $\mathbb{E}\,\|a_{j^*}\|_0 = \Theta(k)$. Next we bound $\mathbb{P}(\neg\mathcal{E})$. For $j \ge j^*$, we know $\mathbb{E}\,\|a_j\|_0 \le 4k/2^{j-j^*}$. Thus, letting μ denote $\mathbb{E}\,\|a_j\|_0$,

$$\mathbb{P}(\|a_j\|_0 > 16k) < \left(\frac{e^{\frac{16k}{\mu}-1}}{(\frac{16k}{\mu})^{\frac{16k}{\mu}}}\right)^{\mu} < \left(\frac{16k}{\mu}\right)^{-\Omega(k)} < (e^{-Ck})^{j-j^*} \tag{12}$$

for some constant $C > 0$ by the Chernoff bound and the fact that $16k/\mu \ge 4 > e$. Recall that the Chernoff bound states that for X a sum of independent Bernoullis,

$$\forall \delta > 0,\ \mathbb{P}(X > (1+\delta)\,\mathbb{E}\,X) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mathbb{E}\,X}.$$

Then by a union bound over $j \ge j^*$ and applying (12),

$$\mathbb{P}(\neg\mathcal{E}) = \mathbb{P}(\exists j \ge j^* : \|a_j\|_0 > 16k) < \sum_{j=j^*}^{\infty} (e^{-Ck})^{j-j^*} = O(e^{-Ck}).$$

Remark 2. As already mentioned, the protocol given above and the one described in [JST11] using $O(k \log^2 n)$ bits differ in minor points. First: the protocol there used $\lfloor \log_2 n \rfloor$ different hash functions h_j, but as seen above, only $\lfloor \log_2(n/k) \rfloor$ are needed. This already improves one $\log n$ factor to $\log(n/k)$. The other improvement comes from replacing the k-sparse recovery structure with $2k$ rows used in [JST11] with our Corollary 4. Note the matrix Π_k in our corollary has even *more* rows, but the key point is that the bit complexity is improved. Whereas using a k-sparse recovery scheme as described in [JST11] would use $2k$ linear measurements of a k-sparse vector $w \in \{-1, 0, 1\}^n$ with $\log n$ bits per measurement (for a total of $O(k \log n)$ bits), we use $O(k \log(n/k))$ measurements with only $O(1)$ bits per measurement. The key insight is that we can work over $\mathbb{F}_3^n$ instead of $\mathbb{R}^n$ when the entries of w are in $\{-1, 0, 1\}$, which leads to our slight improvement.

A.2 Proof of the existence of the desired $\mathcal{S}_{u,m}$

Lemma 7 (restated). For any integers $u \ge 1$ and $1 \le m \le u/(4e)$, there exists a collection $\mathcal{S}_{u,m} \subset \binom{[u]}{m}$ with $\log|\mathcal{S}_{u,m}| = \Theta(m \log(u/m))$ such that for all $S \neq S' \in \mathcal{S}_{u,m}$, $|S \cap S'| < m/2$.

Proof. The proof is via the probabilistic method. We pick $S_1, \ldots, S_N$ independently, each one uniformly at random from $\binom{[u]}{m}$. Fix $i \neq j \in [N]$. Imagine S_i being fixed and picking the m elements of S_j one by one. Let X_k denote the indicator random variable for the event that the kth element picked is also in S_i. Then $|S_i \cap S_j| = \sum_{k=1}^{m} X_k$, and we set $\mu := \mathbb{E}\,|S_i \cap S_j|$, which is m^2/u

by linearity of expectation. We have $\mathbb{P}(|S_i \cap S_j| \geq m/2) = \mathbb{P}(|S_i \cap S_j| \geq (1+\delta)\mu)$ for $\delta = u/(2m) - 1$. The X_k are not independent, but they are negatively dependent. Thus the Chernoff bound yields

$$\mathbb{P}(|S_i \cap S_j| \geq (1+\delta)\mu) \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq \left(\frac{e^{\frac{u}{2m}-1}}{(\frac{u}{2m})^{\frac{u}{2m}}}\right)^{m^2/u} \leq \left(\frac{u}{2em}\right)^{-\frac{m}{2}}.$$

Setting $N = \sqrt{(u/(2em))^{m/2} - 1}$ so that $\binom{N}{2} \leq N^2 = (u/(2em))^{m/2} - 1$, by a union bound with positive probability $|S_i \cap S_j| < m/2$ for all $i \neq j$, simultaneously, as desired. Note for this choice of N, we have $\log |\mathcal{S}_{u,m}| = \log N = \Theta(m \log(u/m))$. ☐

A.3 Another variant of samplers, with applications

In this section we define another variant of the ℓ_0-sampling problem and provide a solution for it. The solution is a minor modification of the ℓ_0-sampling algorithm of [JST11].

Definition 1. *In the turnstile streaming problem* ℓ_0-sampling$_k(\delta_1, \delta_2)$*, there is a vector* $z \in \mathbb{R}^n$ *receiving turnstile streaming updates, and the answer to* `query()` *must behave as follows:*

- *With probability at most δ_1, the output can be "Fail".*
- *With probability at most δ_2, the output can be arbitrary.*
- *Otherwise, the output should be a uniformly random subset of size* $\min\{k, \|z\|_0\}$*, without replacement, from support(z).*

One can define the support-finding$_k(\delta_1, \delta_2)$ problem analogously, as well as similar variants for other desired output distributions. The important distinction is that there are two types of failures, which are allowed to happen with different probabilities.

The description of algorithm given in [AGM12a] for outputting connected components is based on a support-finding subroutine that always knows when it fails and outputs Fail at those times, and [AGM12a] cites [JST11] for implementing this subroutine. Unfortunately, the algorithm of [JST11] does not provide this behavior and can behave arbitrarily when it fails. One can avoid this issue by simply conditioning on never failing, which would require [AGM12a] to call the [JST11] subroutine with $\delta < 1/\mathit{poly}(n)$. This would worsen their claimed space for finding connected components from $O(n \log^3 n)$ bits to $O(n \log^4 n)$ bits. As mentioned in Footnote 2 though, it is apparent from the correctness analysis of the algorithm in [AGM12a] that their algorithm actually only needs a support-finding$_1(\delta_1, \delta_2)$ data structure for δ_1 a small but universal constant, and $\delta_2 = 1/\mathit{poly}(n)$. As we sketch below, ℓ_0-sampling$_k(\delta_1, \delta_2)$ can be solved in the general turnstile model in $O((t \log n + \log(n/\delta_2)) \log(n/t))$ bits of space for $t = \max\{k, \log(1/\delta_1)\}$, based on modifying the algorithm of [JST11]. This leads to an implementation of the connectivity algorithm of [AGM12a] using space $O(n \log^3 n)$ bits as they claim.

We will need the following standard lemma.

Lemma 10. *There is a turnstile streaming algorithm for testing whether a vector $z \in \mathbb{R}^n$ updated in a stream is identically zero. If $z = 0$, the algorithm outputs $z = 0$. If $z \neq 0$, the algorithm outputs $z = 0$ with probability at most δ. The space consumption is $O(\log(nT/\delta))$ bits, assuming that the entries of z are always integers bounded by T in magnitude.*

Proof. We pick a prime p larger than $T + n/\delta$. We then pick a random $r \in \mathbb{F}_p$ at the beginning of the stream, and throughout the stream we maintain a single number in memory: $q(r) = \sum_{i=1}^{n} z_i r^i \bmod p$. That is, during an update "$z_i \leftarrow z_i + \Delta$", we add $\Delta \cdot r^i$ to our counter, where the addition, multiplication, and exponentiation are all in $\mathbb{F}_p$. Note if $z \neq 0$, q is a non-zero degree-n polynomial (since $p > T$) and thus has at most n zeroes in $\mathbb{F}_p$, implying that $\mathbb{P}_r(q(r) = 0) \leq n/p < \delta$. $\square$

We now sketch the modification to the algorithm of [JST11] which solves the above variant of ℓ_0-sampling with the desired space complexity. We assume that the vector z always has integer entries and $\|z\|_\infty \leq T = poly(n)$ throughout the stream. First we describe the algorithm without regard for the space needed to store hash functions. For fixed k and δ_2, we give an algorithm which achieves failure probability $\delta_1 = \exp(-\Theta(k))$ so that $t = \Theta(k)$. The algorithm of [JST11] is then similar to the upper bound for **UR** in Section A.1 (see also [CF14]). Here we set $L = \Theta(\log(n/k))$. We then at each level, as in Section A.1, use a Ck-sparse recovery scheme for vectors in $\mathbb{F}_p^n$ for some prime $p > \max\{n, T\}$ and constant $C > 0$ to arise in the analysis later (so that taking each entry of the recovered vector modulo p still returns the same vector). Such a measurement matrix Π_{Ck} only needs $2Ck$ measurements over $\mathbb{F}_p$ and thus the recovery scheme uses $O(k \log p) = O(k \log n)$ bits. For example, one can use Π being a $2Ck \times n$ Vandermonde matrix, so that no Ck-sparse vector is in its kernel. Recovery can be done by brute force. Alternatively one can use syndrome decoding, which uses the same space but allows recovery in time $poly(k \log p)$ time (see [DORS08, Section E]). Now, suppose level j^* is the level we return our output from, as per the scheme in Algorithm 10. If w_j denotes the output of the recovery scheme, before returning w_j we first would like to check that $w_j = a_j$ with a_j denoting $z|_{h_j^{-1}(1)}$, i.e. that $d_j = w_j - a_j$ is equal to zero. For this we use the zero-tester algorithm of Lemma 10 as a subroutine, with failure probability parameter δ_2. Thus overall we obtain the desired guarantee with space $O((k \log n + \log(n/\delta_2)) \log(n/k))$, as desired. The analysis of correctness is the same as in Section A.1, based on the same events $\mathcal{E}, \mathcal{F}$. The difference is that $\mathcal{E}$ or $\mathcal{F}$ fail to hold, which happens with probability at most δ_1, then if the level j we base our output on does not provide us with a k-sparse vector, then we catch this with the zero-tester and say Fail (except with probability δ_2).

Now we must take into account the space to store the hash functions $h_0, \ldots, h_L$. We implement these hash functions by a single hash function $h : [n] \to [n]$ where we then interpret the event "$h_j(i) = 1$" as occurring when the least significant bit of $h(i)$ is equal to j. We now just need to derandomized the selection of this single hash function. For this, we need to make sure that h is still chosen in such a way that $\mathbb{P}_h(\neg\mathcal{E} \vee \neg\mathcal{F}) < \delta_2$. For the purposes of this algorithm, we modify the definition of $\mathcal{E}$ to be stronger: here we consider it to be the event that $\sum_{j=j^*}^{L} \|a_j\|_0 \leq Ck$ for some constant $C > 0$, i.e. the *total* number of indices from $support(z)$ that hash to levels $j \geq j^*$ is $O(k)$. Then $\mathbb{P}(\neg\mathcal{E}) < \exp(-\Omega(k))$ is still true by a union bound and Chernoff bound as in Section A.1, and $\mathcal{F}$ is not modified from that section, so we also have $\mathbb{P}(\neg\mathcal{F}) < \exp(-\Omega(k))$, as desired. Now, we cannot afford to store a truly random $h : [n] \to [n]$, so we instead determine h using Nisan's pseudorandom generator (PRG) [Nis92]. More specifically, it suffices to fool the following two branching programs up to statistical distance δ_2, one branching program for each of $\mathcal{E}, \mathcal{F}$. For $\mathcal{E}$, we have a branching program that sees the values $h(1), \ldots, h(n)$ in order and maintains a single word of memory which simply counts the number of items $i \in [n]$ that then hash to a level $j \geq j^*$. The space of this branching program is then $\log n$ bits which is certainly at most $S = \log(n/\delta_2)$, and the randomness needed is $R = n \log n$. Thus when using Nisan's PRG, the probability of $\mathcal{E}$ occurring changes by at most an additive $\exp(-\Omega(S)) < \delta_2$, and the required seed length for Nisan's PRG is $O(S \log(R/S)) = O(\log(n/\delta_2) \log n)$ bits. For the next branching program, recall that $\mathcal{F}$ is the event that $\|a_{j^*}\|_0 \geq k$. We thus set up a similar branching program, which sees $h(1), \ldots, h(n)$ in order and counts the number of $i \in [n]$ hashing to level j^*. The seed length and error are the same as for the first branching program. We have thus established the following theorem.

Theorem 7. *For any $k \geq 1$ and $0 < \delta_1, \delta_2 < 1$, there is a solution to the ℓ_0-sampling$_k(\delta_1, \delta_2)$ in turnstile streams with space complexity $O((t \log n + \log(n/\delta_2)) \log(n/t))$ bits, where $t = \max\{k, \log(1/\delta_1)\}$.*