

Optimal lower bounds for universal relation, samplers, and finding duplicates

Jelani Nelson*

Jakub Pachocki†

Zhengyu Wang‡

March 23, 2017

Abstract

In the communication problem **UR** (*universal relation*) [KRW95], Alice and Bob respectively receive x and 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 $\Theta(\min\{n, \log(1/\delta) \log^2(\frac{n}{\log(1/\delta)})\})$ bits for failure probability δ . We also show that for a more general problem **UR_k**, in which the output must be $\min\{k, |\text{support}(x - y)|\}$ distinct indices i such that $x_i \neq y_i$, the optimal randomized one-way communication complexity is $\Theta(\min\{n, t \log^2(n/t)\})$ bits where $t = \max\{k, \log(1/\delta)\}$. Our lower bounds hold even if promised $\text{support}(y) \subset \text{support}(x)$.

As a corollary, we obtain optimal lower bounds for sampling problems in strict turnstile streams, as well as for the problem of finding duplicates in a stream. Specifically, consider a high-dimensional vector $x \in \mathbb{R}^n$ receiving streaming updates of the form “ $x_i \leftarrow x_i + \Delta$ ”, and in response to a query we must with probability $1 - \delta$ recover *any* element $i \in \text{support}(x) = \{j : x_j \neq 0\}$. Our lower bound implies the first optimal $\Omega(\log(1/\delta) \log^2 n)$ -bit space lower bound for any solution to this problem as long as $\delta > 2^{-n^{.99}}$, matching an upper bound of [JST11]. Our result thus implies optimal lower bounds for the so-called “ ℓ_p -sampling” problem for any $0 \leq p < 2$ in the strict turnstile model, as well as variations in which a query response must include $\min\{k, |\text{support}(x)|\}$ elements from $\text{support}(x)$. Our lower bounds also do not need to use large weights, and hold even if it is promised that $x \in \{0, 1\}^n$ at all points in the stream.

More easily explained in the language of the above support-finding turnstile streaming problem, our lower bound operates by showing that any algorithm \mathcal{A} solving that problem 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.

*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.

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 $\Delta \in \{-M, -M + 1, \dots, M\}$. During a query, one must return a uniformly random element from $\text{support}(x) = \{i : z_i \neq 0\}$. 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 algorithms¹, 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 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 $M \leq \text{poly}(n)$ with failure probability δ , which in fact gave $\min\{\|z\|_0, \Theta(\log(1/\delta))\}$ uniform samples from the support of z , using space $O(\log^2 n \log(1/\delta))$ bits (here $\|z\|_0$ denotes $|\text{support}(z)|$). Thus we say their data structure actually solves the harder problem of ℓ_0 -sampling $_{\Theta(\log(1/\delta))}$ 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 $\text{support}(z)$. The upper and lower bounds in [JST11] thus match up to a constant factor for $k = 1$ and δ a constant.

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

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

itself $k = \log n / \log \log n$ times (see [KW90] for a precise definition of composition). If one could prove a strong enough direct sum theorem for communication complexity after composition, such a k -fold composition would yield a function that is provably in \mathbf{P} (and in fact, even in \mathbf{NC}^2), but not in \mathbf{NC}^1 . Proving such a direct sum theorem is still wide open, and the statement that it is true is known as the “KRW conjecture”; see for example the recent works [GMWW14, DM16] toward resolving 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 requires Alice and Bob to agree on the index i only knowing that $x, y \in \{0, 1\}^n$ are unequal. 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 requires an upper bound on the number of rounds [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 $\mathbf{UR}^C, \mathbf{UR}_k^C$ to denote the variants when promised $\text{support}(y) \subset \text{support}(x)$, and also Bob knows $\|x\|_0$. Clearly $\mathbf{UR}, \mathbf{UR}_k$ can only be harder than $\mathbf{UR}^C, \mathbf{UR}_k^C$, 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 particular, if $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(f)$ 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 $\text{FindDuplicate}(n)$ with failure probability δ is at least $\mathbf{R}_{\frac{7}{8} + \frac{\delta}{8}}^{\rightarrow, \text{pub}}(\mathbf{UR})$. In $\text{FindDuplicate}(n)$, 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 $\text{FindDuplicate}(n)$ with failure probability at most $(1 + \delta)/2$ in the same space (and thus the space must be at least $\mathbf{R}_{\frac{15}{16} + \frac{\delta}{16}}^{\rightarrow, \text{pub}}(\mathbf{UR})$). 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 $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}) = \Omega(\log^2 n)$ 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}^C . 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 reduction from \mathbf{UR}^C 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 $\text{support}(z)$, 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 never deleted unless they had previously been inserted, thus only requiring sampler subroutines that are correct with the strict turnstile promise. 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 $\text{FindDuplicate}(n)$ and from \mathbf{UR}_k^{\subset} to support-finding_k . In support-finding_k we must report $\min\{k, \|z\|_0\}$ elements in $\text{support}(z)$. In the claims below, δ is the failure probability for the considered streaming problem.

Claim 1. *Any one-pass streaming algorithm for $\text{FindDuplicate}(n)$ 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 $\text{FindDuplicate}(n)$. Alice creates a stream consisting of all elements of $\text{support}(x)$ 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 $[n] \setminus \text{support}(y)$. Then there must be a duplicate in the resulting concatenated stream, and all duplicates i satisfy $x_i \neq y_i$. \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 $i \in \text{support}(x)$, 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 $i \in \text{support}(y)$ 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 $\text{support}(z)$ satisfying some distributional requirements is only a harder problem than finding *any* $\min\{k, \|z\|_0\}$ elements of $\text{support}(z)$. \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 $\text{FindDuplicate}(n)$, 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 $1 \leq k \leq n$, $\mathbf{R}_{\delta}^{\rightarrow, pub}(\mathbf{UR}_k^{\subset}) = \Theta(\min\{n, t \log^2(n/t)\})$.*

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] was 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 $1/\text{poly}(n)$ (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 as defined above, 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], where they achieved space $\text{poly}(\varepsilon^{-1} \log n)$ for failure probability $1/\text{poly}(n)$. For $1 \leq p \leq 2$ the space was improved to $O(\varepsilon^{-p} \log^3 n)$ bits for constant failure probability [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 $p \neq 1$. 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 describe our proof 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 $O(\min\{n, t \log^2(n/t)\})$ bits. 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). Our lower bound proof in this regime is split into two parts: we show $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}^C) = \Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}})$ and $\mathbf{R}_{\frac{1}{2}}^{\rightarrow, \text{pub}}(\mathbf{UR}_k^C) = \Omega(k \log^2 \frac{n}{k})$ separately. We give an overview the former here, which is the more technically challenging half. Our proof of the latter can be found in Section 4.

We prove the lower bound via an encoding argument. Fix m . A randomized encoder is given a set $S \subset [n]$ with $|S| = m$ and must output an encoding $\text{ENC}(S)$, and a decoder sharing public randomness with the encoder must be able to recover S given only $\text{ENC}(S)$. 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}^C to devise such an encoding/decoding scheme. The encoder pretends to be Alice with input x being the indicator set of S , then lets $\text{ENC}(S)$ 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.

```

1: procedure DEC( $M$ )
2:    $T \leftarrow \emptyset$ 
3:   for  $r = 1, \dots, m$  do
4:     Let  $i$  be Bob's output upon receiving message  $M$  from Alice when Bob's input is  $\mathbf{1}_T$ 
5:      $T \leftarrow T \cup \{i\}$ 
6:   end for
7:   return  $T$ 
8: 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

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

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 $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}^C) = \Omega(\min\{n, \log \binom{n}{m}\}) = \Omega(\min\{n, \log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)}\})$.

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. $\text{ENC}(S)$ 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 $\log n$ bits T would 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:

Lemma 1. *Consider $f: \{0, 1\}^b \times \{0, 1\}^q \rightarrow \{0, 1\}$ 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 random variable Y supported on $\{0, 1\}^q$,*

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

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 $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}^\subset) = \Omega(|R| \log \frac{n}{m}) = \Omega(\min\{n, \log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}}\})$ as desired.

The argument for lower bounding $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}_k^\subset)$ 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 4, 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 $(1 + s + s') \geq \log \binom{n}{m}$. In particular, $s \geq \log \binom{n}{m} - s' - 1$.*

Section 3 provides the full details of the proof that $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}^\subset) = \Omega(\min\{n, \log^2(\frac{n}{\log(1/\delta)}) \log \frac{1}{\delta}\})$. We extend our results in Section 4 to \mathbf{UR}_k^\subset for $k \geq 1$, proving a lower bound of $\Omega(k \log^2(n/k))$ communication even for constant failure probability.

3 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 input is y , Alice sends $\mathbf{Alice}(x)$ to Bob, and Bob outputs $\mathbf{Bob}(\mathbf{Alice}(x), y)$, which with probability at least $1 - \delta$ is in $\text{support}(x - y)$. As mentioned in Section 2, we use \mathcal{P} as a subroutine in a scheme for encoding/decoding elements of $\binom{[n]}{m}$ for $m = \lfloor \sqrt{n \log(1/\delta)} \rfloor$. In this section, we assume $\log \frac{1}{\delta} \leq n/64$, since for larger n we have an $\Omega(n)$ lower bound.

3.1 Encoding/decoding scheme

We now describe our encoding/decoding scheme (ENC, 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 $M \leftarrow \mathbf{Alice}(\mathbf{1}_S)$ as part of its output. Moreover, 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 $s_r \leftarrow \mathbf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$. Let b denote a binary string of length R , where b_r records whether \mathbf{Bob} succeeds in round r . ENC also outputs b . If $s_r \in S_{r-1}$, i.e. $\mathbf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$ 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 $\mathbf{Bob}(M, \mathbf{1}_{C_{r-1}})$. 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.

```
1:  $m \leftarrow \lfloor \sqrt{n \log \frac{1}{\delta}} \rfloor$ 
2:  $K \leftarrow \lfloor \frac{1}{16} \log \frac{1}{\delta} \rfloor$ 
3:  $R \leftarrow \lfloor K \log(m/4K) \rfloor$ 
4: for  $r = 0, \dots, R$  do
5:    $n_r \leftarrow \lfloor m \cdot 2^{-\frac{r}{K}} \rfloor$  ▷  $|S_r| = n_r$ , and we have  $n_r - n_{r+1} \geq 2$ 
6: end for
7: Let  $\pi$  be a random permutation on  $[n]$  ▷ Used to generate  $S_r$  and  $C_r$ 
```

Algorithm 3 Encoder ENC.

```
1: procedure ENC( $S$ )
2:    $M \leftarrow \text{Alice}(\mathbf{1}_S)$ 
3:    $A \leftarrow \emptyset$ 
4:    $S_0 \leftarrow S$ 
5:   for  $r = 1, \dots, R$  do
6:      $s_r \leftarrow \text{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$ 
7:      $S_r \leftarrow S_{r-1}$ 
8:     if  $s_r \in S_{r-1}$  then ▷ i.e. if  $s_r$  is a valid sample
9:        $b_r \leftarrow 1$  ▷  $b$  is a binary string of length  $R$ , indicating if Bob succeeds on round  $r$ 
10:       $A \leftarrow A \cup \{s_r\}$ 
11:       $S_r \leftarrow S_r \setminus \{s_r\}$ 
12:    else
13:       $b_r \leftarrow 0$ 
14:    end if
15:    Remove  $|S_r| - n_r$  elements from  $S_r$  with smallest  $\pi_a$ 's among  $a \in S_r$  ▷ So that  $|S_r| = n_r$ 
16:  end for
17:  return ( $M, S \setminus A, b$ )
18: end procedure
```

Algorithm 4 Decoder DEC.

```
1: procedure DEC( $M, B, b$ )
2:    $A \leftarrow \emptyset$ 
3:    $C_0 \leftarrow \emptyset$ 
4:   for  $r = 1, \dots, R$  do
5:      $C_r \leftarrow C_{r-1}$ 
6:     if  $b_r = 1$  then
7:        $s_r \leftarrow \text{Bob}(M, \mathbf{1}_{C_{r-1}})$  ▷ Invariant:  $C_r = S \setminus S_r$  ( $S_r$  is defined in ENC)
8:        $A \leftarrow A \cup \{s_r\}$ 
9:        $C_r \leftarrow C_r \cup \{s_r\}$ 
10:    end if
11:    Insert  $m - n_r - |C_r|$  items into  $C_r$  with smallest  $\pi_a$ 's among  $a \in B \setminus C_r$ 
12:  end for
13:  return  $B \cup A$ 
14: end procedure
```

3.2 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 $\text{DEC}(\text{ENC}(S)) = S$. 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 UR^C -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. $\text{DEC}(\text{ENC}(S)) = S$.

Proof. We claim that for $r = 0, \dots, R$, $\{S_r, C_r\}$ 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 $r - 1$ ($1 \leq r \leq R$), and we consider round r . On round r , by induction $S \setminus S_{r-1} = C_{r-1}$, 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 $a \in B \setminus C_r$, 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 $B \setminus C_r \subseteq S \setminus C_r = S_r$. Let a^* denote $a \in S_r$ that minimizes π_a . Then $a^* \in B \setminus C_r \subseteq S_r$ (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 $\{\pi_a : a \in B \setminus C_r\}$. 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, $A = \{s_r : b_r = 1, r = 1, \dots, R\}$ are the same in ENC and DEC. We know $A \subseteq S$. Since ENC outputs $S \setminus A$, and DEC outputs $(S \setminus A) \cup A$, we have $\text{DEC}(\text{ENC}(S)) = S$. \square

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

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 $\mathbb{E}(\log \binom{n}{m} - \log \binom{n}{W}) \geq d \log \left(\frac{n}{m} - 1\right)$. \square

Lemma 1 (restated). Consider $f: \{0, 1\}^b \times \{0, 1\}^q \rightarrow \{0, 1\}$ 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}^C -protocol with failure probability at most δ . If S is a fixed set and $T \subset S$, $\mathbb{P}(\text{Bob}(\text{Alice}(\mathbf{1}_S), \mathbf{1}_T) \notin S \setminus T) \leq \frac{I(X; T) + H_2(\delta)}{\log \frac{1}{\delta}}$.*

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

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 ,

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

We have $\mathbb{P}(S_r = T|X = x) = \prod_{i=1}^r \mathbb{P}(T \subseteq S_i | T \subseteq S_{i-1})$. 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 $\frac{\binom{n_{i-1} - n_r - 1}{n_{i-1} - n_i - 1}}{\binom{n_{i-1} - 1}{n_{i-1} - n_i - 1}}$, where the equation achieves when $s_i \in S_{i-1} \setminus T$, and ENC takes a uniformly random subset of $S_{i-1} \setminus \{s_i\}$ of size $n_{i-1} - n_i - 1$, so that the subset does not intersect with T .

Next we can prove

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

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

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

By telescoping,

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

Moreover,

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

By our setting of parameters

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

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

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

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 $j = 2, \dots, r$. 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,

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

Then, we bound the product of the rest terms

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

Multiplying two parts proves the lemma. \square

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

Proof. By Lemma 3, the success probability of protocol (ENC, DEC) is 1. By Lemma 2, we have $s \geq \log \binom{n}{m} - s' - 1$, where $s' = \log n + R + \mathbb{E}(\log \binom{n}{|B|})$. 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 $s \geq \frac{R}{3} \log \frac{n}{\log(1/\delta)} - (\log n + 1) = \Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log(1/\delta)})$. \square

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

In this section, we prove the lower bound $\mathbf{R}_{1/2}^{\rightarrow, pub}(\mathbf{UR}_k^{\subset}) = \Omega(\min\{n, k \log^2 \frac{n}{k}\})$. 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 $\text{Alice}_k(x)$ to Bob, and Bob outputs $\text{Bob}_k(\text{Alice}_k(x), y)$. We consider the following encoding/decoding scheme $(\text{ENC}_k, \text{DEC}_k)$ for $S \in \binom{[n]}{m}$. ENC_k computes $M \leftarrow \text{Alice}_k(\mathbf{1}_S)$ 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 $S_0 = S \supseteq S_1 \supseteq \dots \supseteq S_R$ where S_r is generated by sub-sampling each element in S_{r-1} with probability $\frac{1}{2}$. In round r ($r = 1, \dots, R$), ENC_k tries to obtain k elements from S_{r-1} by invoking $\text{Bob}_k(M, \mathbf{1}_{S_{r-1}})$, 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.

4.1 Encoding/decoding scheme

Algorithm 5 Variables Shared by Encoder ENC_k and Decoder DEC_k .

```

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

```

Algorithm 6 Encoder ENC_k .

```

1: procedure  $\text{ENC}_k(S)$ 
2:    $M \leftarrow \text{Alice}_k(\mathbf{1}_S)$ 
3:    $A \leftarrow \emptyset$ 
4:   for  $r = 1, \dots, R$  do
5:      $A_r \leftarrow \text{Bob}_k(M, \mathbf{1}_{S \setminus (S \cap T_{r-1})})$ 
6:     if  $A_r \subseteq S \cap T_{r-1}$  then ▷ i.e. if  $A_r$  is valid
7:        $b_r \leftarrow 1$  ▷  $b$  is a binary string of length  $R$ , indicating if  $\text{Bob}_k$  succeeds in round  $r$ 
8:        $A \leftarrow A \cup (A_r \cap (T_{r-1} \setminus T_r))$ 
9:     else
10:       $b_r \leftarrow 0$ 
11:     end if
12:   end for
13:   return  $(M, S \setminus A, b)$ 
14: end procedure

```

Algorithm 7 Decoder DEC_k .

```
1: procedure  $\text{DEC}_k(M, B, b)$ 
2:    $A \leftarrow \emptyset$ 
3:    $C_0 \leftarrow \emptyset$ 
4:   for  $r = 1, \dots, R$  do
5:      $C_r \leftarrow C_{r-1}$ 
6:     if  $b_r = 1$  then
7:        $A_r \leftarrow \text{Bob}_k(M, \mathbf{1}_{C_{r-1}})$  ▷ Invariant:  $C_r = S \setminus (S \cap T_r)$ 
8:        $A \leftarrow A \cup (A_r \cap (T_{r-1} \setminus T_r))$ 
9:        $C_r \leftarrow C_r \cup (A_r \cap (T_{r-1} \setminus T_r))$ 
10:    end if
11:     $C_r \leftarrow C_r \cup (B \cap (T_{r-1} \setminus T_r))$ 
12:  end for
13:  return  $B \cup A$ 
14: end procedure
```

4.2 Analysis

Theorem 3. $\mathbf{R}_\delta^{\rightarrow, \text{pub}}(\mathbf{UR}_k^{\subset}) = \Omega(k \log^2 \frac{n}{k})$, given that $1 \leq k \leq \frac{n}{2^{10}}$ and $\delta \leq \frac{1}{2}$.

Proof. Let $S_r = S \cap T_r$. Let **SUCC** denote the event that $|S \cap T_R| = |S_R| \geq k$. Note that $\mathbb{E}|S_R| = \frac{1}{2^R}m = 4k$. By the Chernoff bound, $\mathbb{P}(\text{SUCC}) \geq \frac{1}{2}$. 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 $(\text{ENC}_k, \text{DEC}_k)$ always succeeds. By Lemma 2, we have $s \geq \log \binom{n}{m} - s' - 2$, where $s' = \log n + R + \mathbb{E} \log \binom{n}{|B|}$. The size of B is $|B| = |S| - \sum_{r=1}^R (b_r \cdot |A_r \cap (S_{r-1} \setminus S_r)|)$. The randomness used by \mathcal{P} is independent from $S \setminus S_{r-1}$ for every $r \in [R]$. Therefore, $\mathbb{E} b_r \geq 1 - \delta \geq \frac{1}{2}$, and b_r is independent from $|A_r \cap (S_{r-1} \setminus S_r)|$. We have $\mathbb{E}|A_r \cap (S_{r-1} \setminus S_r)| = \frac{k}{2}$, and thus $\mathbb{E}(|S| - |B|) \geq \frac{kR}{4}$. By Lemma 4, $\log \binom{n}{m} - \mathbb{E} \log \binom{n}{|B|} \geq \frac{kR}{4} \cdot \log(\frac{n}{m} - 1) \geq \frac{kR}{9} \log(\frac{n}{k})$. Moreover, $\frac{kR}{10} \log \frac{n}{k} \geq R$. Thus we have $s = \Omega(kR \log \frac{n}{k}) = \Omega(k \log^2 \frac{n}{k})$. \square

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 31st 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 16th 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 28th 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 27th 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 52nd 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 47th Annual ACM 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 23rd 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 27th 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 26th 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 15th 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.
- [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.
- [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 L_p 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.
- [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.
- [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.

- [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 $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}_k)$

In [JST11, Proposition 1] it is shown that $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}_k) = O(\min\{n, t \log^2 n\})$ for $t = \max\{k, \log(1/\delta)\}$. Here we show that a minor modification of their protocol in fact shows the correct complexity $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}_k) = O(\min\{n, t \log^2(n/t)\})$, 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 $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}_k) \leq n$ is obvious: Alice can simply send the message $M = x$, and Bob can then succeed with failure probability 0. We thus now show $\mathbf{R}_{e^{-ck}}^{\rightarrow, pub}(\mathbf{UR}_k) \leq k \log^2(n/k)$ for some constant $c > 0$, which completes the proof of the upper bound. We assume $k \leq n/2$ (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 1. We first need the following lemma.

Lemma 7. *Let \mathbb{F}_q be a finite field and $n > 1$ an integer. Then for any $1 \leq k \leq \frac{n}{2}$, 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 $\|w\|_0, \|w'\|_0 \leq k$, $\Pi_k w \neq \Pi_k w'$.*

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 $\|v\|_0 \leq 2k$. 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 $\|v\|_0 \leq 2k$ is at most $\binom{n}{2k} \cdot q^{2k}$. For any fixed v , $\mathbb{P}(\Pi_k v = 0) = q^{-m}$. Thus

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

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

Corollary 2. Let \mathbb{F}_q be a finite field and $n > 1$ an integer. Then for any $1 \leq k \leq \frac{n}{2}$, 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 $\|w\|_0 \leq k$, $\mathcal{R}(\Pi_k w) = w$.

Proof. Given Lemma 7, a simple such \mathcal{R} is as follows. Given some $y = \Pi_k w^*$ with $\|w^*\|_0 \leq k$, \mathcal{R} loops over all w in \mathbb{F}_q^n with $\|w\|_0 \leq k$ 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 $h_0, \dots, h_L : [n] \rightarrow \{0, 1\}$ 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 2 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 $j = 0, \dots, L$ where $v_j \in \mathbb{F}_q^m$, and her message to Bob is $M = (v_0, \dots, v_L)$. 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 8 Bob's algorithm in the \mathbf{UR}_k protocol.

```

1: procedure BOB( $v_0, \dots, v_L$ )
2:   for  $j = L, L - 1, \dots, 0$  do
3:      $v_j \leftarrow v_j - \Pi_{16k} y|_{h_j^{-1}(1)}$ 
4:      $w_j \leftarrow \mathcal{R}(v_j)$ 
5:     if  $\|w_j\|_0 \geq k$  or  $j = 0$  then
6:       return an arbitrary  $\min\{k, \|w_j\|_0\}$  elements from  $\text{support}(w_j)$ 
7:     end if
8:   end for
9: 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 $0 \leq \|a\|_0 \leq n$, there either (1) exists a unique $0 \leq j^* \leq L$ such that $2k \leq \mathbb{E} \|a_j\|_0 \cdot 2^{-j^*} < 4k$, or (2) $\|a\|_0 < 2k$ (in which case we define $j^* = 0$). Let \mathcal{E} be the event that $\|a_j\|_0 \leq 16k$ simultaneously for all $j \leq j^*$. Let \mathcal{F} be the event that *either* we are in case (2), or we are in case (1) and $\|a_{j^*}\|_0 \geq k$ holds. Note that conditioned on \mathcal{E}, \mathcal{F} both occurring, Bob succeeds by Corollary 2.

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 $\mathbb{P}(\neg \mathcal{F}) \leq \mathbb{P}(\|a_{j^*}\|_0 < \frac{1}{2} \cdot \mathbb{E} \|a_{j^*}\|_0)$, 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 $j \geq j^*$, we know $\mathbb{E} \|a_j\|_0 \leq 4k/2^{j-j^*}$. Thus, letting μ denote $\mathbb{E} \|a_j\|_0$,

$$\mathbb{P}(\|a_j\|_0 > 16k) < \left(\frac{e^{\frac{16k}{\mu}} - 1}{\frac{16k}{\mu}} \right)^\mu < \left(\frac{16k}{\mu} \right)^{-\Omega(k)} < (e^{-Ck})^{j-j^*} \quad (10)$$

for some constant $C > 0$ by the Chernoff bound and the fact that $16k/\mu \geq 4 > e$. 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 $j \geq j^*$ and applying (10),

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

Remark 1. 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 2. 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.