We sincerely thank all the reviewers for their thorough feedback, and detailed comments. We will incorporate the feedback regarding presentation of the paper, including adding definitions for geometric distribution and cryptographic pseudorandom functions, adding more motivations and intuition about the proposed algorithm, and fixing typos.

**General comments:**

- Regarding the comments about *empirical comparison with prior work* (mentioned by **Reviewers 3 and 4**), we agree that an empirical comparison with prior work would be desirable, and we will incorporate a comparison to prior work ([32] and [36] specifically) in future versions of the paper. We focused initially on analytical comparisons because empirical accuracy can depend on various optimizations of the estimators used on the underlying sketches [24].
- Regarding the comments about *update time complexity of our algorithm* (mentioned by **Reviewers 2 and 3**), we first want to apologize for an error in quoting the update time of related work [32] in Table 1. The correct time complexity is $O(1/\gamma^2)$ (instead of poly$(\ln u)$) which is the same as that of our algorithm. (Analogous to our $m$ (# of hashes), in Algorithm 1 in [32], the term $r = O(1/\gamma^2)$ governs their update time.) On a related note, we will clarify that [36] assumes the input is a set rather than a multiset, i.e., it assumes there are no duplicates in the stream. (This assumption is crucially used in their sketch update.) We will add more detail regarding this in the revision.
- Regarding *empirically measuring the running time* (mentioned by **Reviewer 4**): running time is heavily dependent on hardware and optimizations (e.g., serial vs parallel updates of the $m$ estimates in our Algorithm 3). Furthermore, since this is a streaming algorithm, the overall running time is dependent how the stream processor is implemented. A comparison of optimized running times is beyond the scope of our work.
- *Concurrent work:* We recently became aware of concurrent work of Choi, Dachman-Soled, Kulkarni, and Yerukhimovich (published after the NeurIPS submission deadline, in PETS 2020). Our algorithm offers better accuracy than theirs for comparable privacy guarantees. We will include a discussion of their work in future versions.

**Reviewer 1**:

- *The new algorithm depends on the assumption that there exists an ideal uniform random hash function and it has neglected the cost of storing this function. Although this assumption can be removed, the space has boosted from $O(\log \log u)$ to $O(\log u + \log m)$, which is bad*: Indeed, the space required goes up when we remove the ideal hash function assumption. However, even then the space/accuracy trade-off is at least polynomially better than the prior work. We will address this point better in our revision.
- *I am not sure whether the algorithm can work under pure DP (it seems not). I do not think it is fair to compare this work with the other literature in table 1, which have pure DP guarantees:* Good question. We do not know whether our result can be achieved with pure DP (i.e. $\delta = 0$) or, for that matter, without the pseudorandom hash functions. Since algorithms making different assumptions are not directly comparable, we explicitly stated the assumptions made by each algorithm in Table 1. We will try to clarify the table further.

**Reviewer 2**:

- *In Row 62, authors choose $\beta$ to $1/3$, which is not quite small:* The choice of the failure probability $\beta = 1/3$ was only for the ease of the reader to ignore the $\sqrt{\log(1/\beta)}$ term while interpreting space bounds. In fact, $\beta$ can be any small constant value. In the experiments, since we explicitly choose $m$ (# of hash functions), $\beta$ does not play any role there.

**Reviewer 3**:

- *What is the meaning of $\omega$?* That is a typo. We meant $k$ instead of $\omega$ in the abstract, and have corrected it.
- *The time complexity is $O(1/\gamma^2)$ per update which may be large in real application compare to $O(\ln u)$ in Darakhshan etc. For quantile estimators we need the $\gamma$ to be small enough, so the time complexity would be large. The authors do not provide the running time of their algorithms in their experiments:* Please see the general comments at the top for more detail about empirical time measurement and comparison to the update time in [32]. In the experiments, although we used a slightly different basic sketch for the quantile estimator, the parameter $m$ — the number of copies of the basic sketch (Algorithm 1) — is the same for all of three estimators. Therefore, the empirical running time is the same for the three estimators.

**Reviewer 4**:

- *Adding a chart showing how the bounds are behaving when tuning the different parameters regarding time and space complexities:* Thank you for the suggestion. We will add a chart comparing the analytical bounds on space and update time across various choice of parameters.