The required revisions are:

1. Commit to what will be released for reproducibility. This includes not only the attack code but the evaluation scripts used to produce the results.

- ✓ We added a footnote in Section 7 that links to our GitHub repository. This repository contains our attacks' scripts along with a README that explains how one can reproduce our results.

2. Include in the paper the motivation for stating performance costs in terms of the cost of hashing vs. PRF that you gave in the rebuttal.

- ✓ We have added a short explanation that hash/PRF execution time dominates HLL insertion cost in Section 8.6.

3. Formal model and security proofs. a) Fix hole in security proof noted by Reviewer B. b) rewrite formal model and proof for clarity, in line with what Reviewer E and others asked. In particular, make sure that necessary features of the model and proof are there and not in the "intuition" section.

The formal security model is fully detailed in Section 8.3, while Section 8.2 is only included to provide intuition.

There is no "hole" in our security proof, but we understand where Reviewer B is coming from here.

As we state in Section 8.3, when defining the insert(.) oracle, "Since HLL and variants of it we consider here have the property that reinserting an item x does not change the internal state of the HLL sketch (nor the cardinality estimate computed from it) we assume without loss of generality that all queries x made by A to insert(·) are distinct."

Such repeated queries are sometimes called "redundant queries" or "useless queries" because they do not tell the adversary anything it did not already know. A good example can be found on page 6 of Rogaway-Shrimpton, "Deterministic Authenticated-Encryption", https://web.cs.ucdavis.edu/~rogaway/papers/keywrap.pdf: "We assume that the adversary [...] does not ask left queries outside of H×X ; and does not repeat a query. The last two assumptions are without loss of generality, as an adversary that violated any of these constraints could be replaced by a more efficient and equally effective adversary [...] that did not."

Now, under the assumption of having distinct insert(.) queries, all the queries made to the function F in our existing proof (in either real or ideal worlds) ARE distinct and the ideal world version of F can be implemented just by selecting random values for each query (as we do in the proof). Of course, as the reviewer correctly notes, if the insert(.) queries were NOT distinct, this random sampling approach to simulating F in the ideal world would lead to distinguishable behaviour, and the proof would not go through.

An alternative approach to the proof goes as follows. We do allow the adversary to make repeated insertion queries, and inside the proof, we build a table of values (x_i,y_i) such that y_i = F(x_i), where in the ideal world each y_i is randomly sampled, but we make sure to use the same y_i value each time the same x_i value is used (so we use the table to maintain consistency of replies). This additional "book-keeping" is also a standard approach for proofs involving PRFs when repeated queries are not (or cannot be) eliminated. It ends up at the same place as our existing proof (the theorem statement would now set q to the number of **distinct** x_i inserted). But it's longer and less elegant because of the additional book-keeping that is needed.

We hope the above discussion convinces the reviewer that our "without loss of generality" statement is justified and that our existing proof is good. We did carefully consider just switching to the alternative approach outlined above, but we prefer the more elegant approach resulting from recognising and eliminating redundant queries.

4. Add more experiments with non-empty sketches to show the intuition that attacks would get easier with fuller sketches (as promised in the rebuttal to Reviewer C's comment).

- ✓ We have extended the theoretical analysis of non-empty sketches in scenario S2 in Section 5.3.1 to show how fuller sketches make later adversarial insertions easier. The analysis in Section 5.3.2 actually already showed this for scenario S4.
- ✓ We improved how the experimental results are presented, by gathering the attacks on empty-sketches into a single table (Table 2), and adding plots that show how the attacks on non-empty sketches improve as the attacked sketch gets fuller (Figures 2 and 3). We reduced the experiments in the setting of [RT20] to the S1 scenario, focussing on giving results that are directly comparable to [RT20] (see Table 3). We also added the number of resets. In case of misunderstanding, we'd just like to point to Reviewer C that resets are only needed in S1.
- ✓ We added extra experiments specific to the Redis implementation, see the text in Section 7.2, as well as Figure 4 and Table 4.
- ✓ Additionally, we increased the number of iterations in all our experiments from 30 to 50, and changed the number of targeted buckets from 100 to m/2 to be consistent with the parameters mentioned in the attack section.

Unfortunately, the diff tool is quite unhelpful regarding showing differences in tables; we thus direct the reviewers to the updated pdf to view the updated results and formatting.

5. Add a discussion of potential harms as promised in the rebuttal.

- ✓ We have added a short discussion on harms and mitigations in a new Section 1.1 in the introduction. We have also contacted the Redis core developers to notify them of our results; they have responded saying they plan to evaluate our work but they are currently fully occupied with preparation for a major release.

6. Address reviewers' comments on the writing.

- ✓ We have fixed the sectioning in the appendix.

- ✓ We have fixed all typos identified by the reviewers.
- ✓ We have made numerous small improvements in the writing, as suggested by reviewers. These should be visible in the diff file.
- ✓ To reviewer C: in §5.2.2 we do mean B.