Damien Desfontaines, Andreas Lochbihler, and David Basin

Cardinality Estimators do not Preserve Privacy

Abstract: Cardinality estimators like HyperLogLog are sketching algorithms that estimate the number of distinct elements in a large multiset. Their use in privacy-sensitive contexts raises the question of whether they leak private information. In particular, can they provide any privacy guarantees while preserving their strong aggregation properties?

We formulate an abstract notion of cardinality estimators, that captures this aggregation requirement: one can merge sketches without losing precision. We propose an attacker model and a corresponding privacy definition, strictly weaker than differential privacy: we assume that the attacker has no prior knowledge of the data. We then show that if a cardinality estimator satisfies this definition, then it cannot have a reasonable level of accuracy. We prove similar results for weaker versions of our definition, and analyze the privacy of existing algorithms, showing that their average privacy loss is significant, even for multisets with large cardinalities. We conclude that strong aggregation requirements are incompatible with any reasonable definition of privacy, and that cardinality estimators should be considered as sensitive as raw data. We also propose risk mitigation strategies for their real-world applications.

1 Introduction

Many data analysis applications must count the number of distinct elements in a large stream with repetitions. These applications include network monitoring [22], online analytical processing [35, 40], query processing, and database management [42]. A variety of algorithms, which we call cardinality estimators, have been devel-

Damien Desfontaines: ETH Zurich / Google, damien@desfontain.es
Andreas Lochbihler: Digital Asset, mail@andreas-lochbihler.de
David Basin: ETH Zurich, basin@inf.ethz.ch

oped to do this efficiently, with a small memory footprint. These algorithms include PCSA [25], LogLog [16], and HyperLogLog [24]. They can all be parallelized and implemented using frameworks like MapReduce [13]. Indeed, their internal memory state, called a *sketch*, can be saved, and sketches from different data shards can be aggregated without information loss.

Using cardinality estimators, data owners can compute and store sketches over fine-grained data ranges, for example, daily. Data analysts or tools can then *merge* (or *aggregate*) existing sketches, which enables the subsequent estimation of the total number of distinct elements over arbitrary time ranges. This can be done without re-computing the entire sketch, or even accessing the original data.

Among cardinality estimators, HyperLogLog [24] and its variant HyperLogLog++ [29] are widely used in practical data processing and analysis tasks. Implementations exist for many widely used frameworks, including Apache Spark [1], Google BigQuery [2], Microsoft SQL Server [3], and PostgreSQL [4]. The data these programs process is often sensitive. For example, they might estimate the number of distinct IP addresses that connect to a server [41], the number of distinct users that perform a particular action [5], or the number of distinct devices that appeared in a physical location [34]. To illustrate this point, we present an example of a location-based service, which we will use throughout the paper.

Example 1. A location-based service gathers data about the places visited by the service's users. For each place and day, the service stores a sketch counting the identifiers of the users who visited the place that day. This allows the service's owners to compute useful statistics. For example, a data analyst can merge the sketches corresponding to the restaurants in a neighborhood over a month, and estimate how many distinct individuals visited a given restaurant during that month. The cost of such an analysis is proportional to the number of aggregated sketches. If the queries used raw data instead, then each query would require a pass over the entire dataset, which would be much more costly.

Note that the type of analysis to be carried out by the data analyst may not be known in advance. The data analysts should be able to aggregate arbitrarily many sketches, across arbitrary dimensions such as time

or space. The relative precision of cardinality estimation should not degrade as sketches are aggregated. □

Fine-grained location data is inherently sensitive: it is extremely re-identifiable [26], and knowing an individual's location can reveal private information. For example, it can reveal medical conditions (from visits to specialized clinics), financial information (from frequent visits to short-term loan shops), relationships (from regular co-presence), sexual orientation (from visits to LGBT community spaces), etc. Thus, knowing where a person has been reveals sensitive information about them.

As the above example suggests, an organization storing and processing location data should implement risk mitigation techniques, such as encryption, access controls, and access audits. The question arises: how should the sketches be protected? Could they be considered sufficiently aggregated to warrant weaker security requirements than the raw data?

To answer this question, we model a setting where a data owner stores sketches for cardinality estimation in a database. The attacker can access some of the stored sketches and any user statistics published, but not the raw data itself. This attacker model captures the *insider risk* associated with personal data collections where insiders of the service provider could gain direct access to the sketch database. In this paper, we use this insider risk scenario as the default attacker model when we refer to the attacker. In the discussion, we also consider a weaker attacker model, modeling an *external attacker* that accesses sketches via an analytics service provided by the data owner. In both cases, we assume that the attacker knows the cardinality estimator's internals.

The attacker's goal is to infer whether some user is in the raw data used to compute one of the accessible sketches. That is, she picks a *target user*, she chooses a sketch built from a stream of users, and she must guess whether her target is in this stream. The attacker has some *prior knowledge* of whether her target is in the stream, and examining the sketch gives her a *posterior knowledge*. The increase from prior to posterior knowledge determines her knowledge gain.

Consider Example 1. The attacker could be an employee trying to determine whether her partner visited a certain restaurant on a given day, or saw a medical practitioner. The attacker might initially have some suspicion about this (the prior knowledge), and looking at the sketches might increase this suspicion. A small, bounded difference of this suspicion might be deemed to be acceptable, but we do not want the attacker to be able to increase her knowledge too much.

We show that for all cardinality estimators that satisfy our aggregation requirement, sketches are almost as sensitive as raw data. Indeed, in this attacker model, the attacker can gain significant knowledge about the target by looking at the sketch. Our results are a natural consequence of the aggregation properties: to aggregate sketches without counting the same user twice, they must contain information about which users were previously added. The attacker can use this information, even if she does not know any other users in the sketch. Furthermore, adding noise either violates the aggregation property, or has no influence on the success of the attack. Thus, it is pointless to try and design privacy-preserving cardinality estimators: privacy and accurate aggregation are fundamentally incompatible.

To show the applicability of our analysis to real-world cardinality estimators, we quantify the privacy of HyperLogLog, the most widely used cardinality estimator algorithm. We show that for common parameters, the privacy loss is significant for most users. Consider, for example, a user A among the 500 users associated with the largest privacy loss, in a sketch that contains 1000 distinct users. An attacker with an initial suspicion of 1% that A is in the sketch can raise her level of certainty to over 31% after observing the sketch. If her initial estimate is 10%, it will end up more than 83%. If her prior is 50%, then her posterior is as high as 98%.

Our main contributions are:

1. We formally define a class of algorithms, which we call *cardinality estimators*, that count the number of distinct elements in a stream with repetitions, and can be aggregated arbitrarily many times (Section 3).
2. We give a definition of privacy that is well-suited to how cardinality estimators are used in practice (Section 4.2).
3. We prove that a cardinality estimator cannot satisfy this privacy property while maintaining good accuracy asymptotically (Section 5). We show similar results for weaker versions of our initial definition (Section 6).
4. We highlight the consequences of our results on practical applications of cardinality estimators (Section 7) and we propose risk mitigation techniques (Section 8).

2 Previous work

Prior work on privacy-preserving cardinality estimators has been primarily focused on *distributed* user counting, for example to compute user statistics for anonymity networks like Tor. Each party is usually assumed to hold a set X_i of data, or a sketch built from X_i, and the goal is to compute the cardinality of $\bigcup_i X_i$, without allowing the party i to get too much information on the sets X_j with $j \neq i$. The attacker model presumes honest-but-curious adversaries.

Tschorsch and Scheuermann [41] proposed a noise-adding mechanism for use in such a distributed context. In [32], each party encrypts their sketch, and sends it encrypted to a tally process, which aggregates them using homomorphic encryption. Ashok et al. [5] propose a multiparty computation protocol based on Bloom filters to estimate cardinality without the need for homomorphic encryption, while Egert et al. [20] show that Ashok et al.'s approach is vulnerable to attacks and propose a more secure variant of the protocol.

Our attacker model, based on insider risk, is fundamentally different to previously considered models: the same party is assumed to have access to a *large number* of sketches; they must be able to aggregate them and get good estimates.

Our privacy definition for cardinality estimators is inspired from differential privacy, first proposed by Dwork et al. [18]. Data-generating probability distributions were considered as a source of uncertainty in [7, 11, 15, 28, 30, 37, 43]; and some deterministic algorithms, which do not add noise, have been shown to preserve privacy under this assumption [7, 11, 28]. We explain in Section 4.2 why we need a custom privacy definition for our setup and how it relates to differential privacy [17] and Pufferfish privacy [30].

Our setting has some superficial similarities to the local differential privacy model, where the data aggregator is assumed to be the attacker. This model is often used to develop privacy-preserving systems to gather statistics [8, 9, 21, 23]. However, the setting and constraints of our work differ fundamentally from these protocols. In local differential privacy, each individual sends data to the server only once, and there is no need for deduplication or intermediate data storage. In our work, the need for intermediate sketches and unique counting leads to the impossibility result.

3 Cardinality estimators

In this section, we formally define cardinality estimators, and prove some of their basic properties.

Cardinality estimators estimate the number of distinct elements in a stream. The internal state of a cardinality estimator is called a *sketch*. Given a sketch, one can estimate the number of distinct elements that have been added to it (the *cardinality*).

Cardinality estimator sketches can also be aggregated: two sketches can be *merged* to produce another sketch, from which we can estimate the total number of distinct elements in the given sketches. This aggregation property makes sketch computation and aggregation embarrassingly parallel. The order and the number of aggregation steps do not change the final result, so cardinality estimation can be parallelized using frameworks like MapReduce.

We now formalize the concept of a cardinality estimator. The elements of the multiset are assumed to belong to a large but finite set $\mathcal{U}$ (the *universe*).

Definition 2. A *deterministic cardinality estimator* is a tuple $\langle M_\varnothing, add, estimate \rangle$, where
- $M_\varnothing$ is the *empty sketch*;
- $add(M, e)$ is the deterministic operation that adds the element e to the sketch M and returns an updated sketch;
- $estimate(M)$ estimates the number of distinct elements that have been added to the sketch.

Furthermore, the *add* operation must satisfy the following axioms for all sketches M and elements $e, e_1, e_2 \in \mathcal{U}$:

$$add(add(M, e), e) = add(M, e) \qquad \text{(idempotent)}$$
$$add(add(M, e_1), e_2) = add(add(M, e_2), e_1) \quad \text{(commutative)}$$

These axioms state that *add* ignores duplicates and that the order in which elements are added is immaterial. Ignoring duplicates is a natural requirement for cardinality estimators. Ignoring order is required for this operation to be used in frameworks like MapReduce, or open-source equivalents like Hadoop or Apache Beam. Since handling large-scale datasets typically requires using such frameworks, we consider commutativity to be a hard requirement for cardinality estimators.

We denote by $M_{e_1, \ldots, e_n}$ the sketch obtained by adding $e_1, \ldots, e_n$ successively to $M_\varnothing$, and we denote by $\mathcal{M}$ the set of all sketches that can be obtained inductively from $M_\varnothing$ by adding elements from any subset of $\mathcal{U}$ in any order. Note that $\mathcal{M}$ is finite and of cardinality at most $2^{|\mathcal{U}|}$. Order and multiplicity do not influ-

ence sketches: we denote by M_E the sketch obtained by adding all elements of a set E (in any order) to $M_\varnothing$.

Later, in Example 7, we give several examples of deterministic cardinality estimators, all of which satisfy Definition 2.

Lemma 3. $add\left(M_{e_1,\dots,e_n}, e_i\right) = M_{e_1,\dots,e_n}$ *for all* $i \leq n$.

Proof. This follows directly from Properties 1 and 2. □

In practice, cardinality estimators also have a *merge* operation. We do not explicitly require the existence of this operation, since *add*'s idempotence and commutativity ensure its existence as follows.

Definition 4. To merge two sketches M and M', choose some $E = \{e_1, \dots, e_n\} \subseteq \mathcal{U}$ such that $M' = M_E$. We define $merge(M, M')$ to be the sketch obtained after adding all elements of E successively to M.

Note that this construction is not computationally tractable, even though in practical scenarios, the *merge* operation must be fast. This efficiency requirement is not necessary for any of our results, so we do not explicitly require it either.

Lemma 5. *The merge operation in Definition 4 is well-defined, i.e., it does not depend on the choice of E. Furthermore, the merge operation is a commutative, idempotent monoid on $\mathcal{M}$ with $M_\varnothing$ as neutral element.*

The proof is given in Appendix A. Note that these properties of the merge operation are important for cardinality estimators: when aggregating different sketches, we must ensure that the result is the same no matter in which order the sketches are aggregated.

Existing cardinality estimators also satisfy efficiency requirements: they have a low memory footprint, and *add* and *merge* run in constant time. These additional properties are not needed for our results, so we omit them in our definition.

We now define *precise* cardinality estimators.

Definition 6. Let E be a set of cardinality n taken uniformly at random in $\mathcal{U}$. The quality of a cardinality estimation algorithm is given by two metrics:

1. Its *bias* $\mathbb{E}\left[n - estimate\left(M_E\right)\right]$
2. Its *variance*

$$\mathbb{V}_n = \mathbb{E}\left[\left(\mathbb{E}\left[estimate\left(M_E\right)\right] - estimate\left(M_E\right)\right)^2\right].$$

Cardinality estimators used in practice are asymptotically unbiased: $\frac{1}{n} \cdot \mathbb{E}\left[n - estimate\left(M_E\right)\right] = o\left(1\right)$. In the rest of this work, we assume that all cardinality estimators we consider are perfectly unbiased, so $\mathbb{V}_n = \mathbb{E}\left[\left(n - estimate\left(M_E\right)\right)^2\right]$. Cardinality estimators are often compared by their *relative standard error* (RSE), which is given by $\sqrt{\mathbb{E}\left[\left(\frac{n - estimate(M_E)}{n}\right)^2\right]} = \frac{\sqrt{\mathbb{V}_n}}{n}$.

A cardinality estimator is said to be *precise* if it is asymptotically unbiased and its relative standard error is bounded by a constant. In practice, we want the relative standard error to be less than a reasonably small constant, for example less than 10%.

Example 7. We give a few examples of cardinality estimators, with their *memory usage* in bits ($m = \log_2 |\mathcal{M}|$) and their relative standard error. As a first step, they all apply a hash function to the user identifiers. Conceptually, this step assigns to all users probabilistically a random bitstring of length 32 or 64 bits. Since hashing is deterministic, all occurrences of a user are mapped to the same bitstring.

– K-Minimum Values [6], with parameter k, maintains a list of the k smallest hashes that have been added to the sketch. With 32-bit hashes, it has a memory usage of $m = 32 \cdot k$, and its unbiased estimator has a RSE of approximately $\frac{1}{\sqrt{k}} \simeq \frac{5.66}{\sqrt{m}}$ [10].

– Probabilistic Counting with Stochastic Averaging (PCSA, also known as FM-sketches) [25] maintains a list of k *bit arrays*. When an element is added to an FM-sketch, its hash is split into two parts. The first part determines which bit array is modified. The second part determines which bit is flipped to 1, depending on the number of consecutive zeroes at the beginning. With k registers and 32-bit hashes, its memory usage is $m = 32 \cdot k$, and its RSE is approximately $\frac{0.78}{\sqrt{k}} \simeq \frac{4.41}{\sqrt{m}}$.

– LogLog [16] maintains a tuple of k registers. Like PCSA, it uses the first part of the hash to pick which register to modify. Then, each registers stores the *maximum* number of consecutive zeroes observed so far. Using $k \geq 64$ registers and a 32-bit hash, its memory usage is $m = 5 \cdot k$ bits, and its standard error is approximately $\frac{1.30}{\sqrt{k}} \simeq \frac{2.91}{\sqrt{m}}$.

– HyperLogLog [29] has the same structure and add operation as LogLog, only its estimate operation is different. Using $k \geq 16$ registers and a 64-bit hash, it has a memory usage of $m = 6 \cdot k$ bits, and its standard error is approximately $\frac{1.04}{\sqrt{k}} \simeq \frac{2.55}{\sqrt{m}}$.

– Bloom filters can also be used for cardinality estimation [36]. However, we could not find an expression of the standard error for a given memory usage in the literature.

All these cardinality estimators have null or negligible ($\ll 1$) bias. Thus, their variance is equal to their mean squared standard error. So the first four are precise, whereas we do not know if Bloom filters are. $\square$

All examples above are deterministic cardinality estimators. For them and other deterministic cardinality estimators, bias and variance only come from the randomness in the algorithm's inputs. We now define *probabilistic* cardinality estimators. Intuitively, these are algorithms that retain all the useful properties of deterministic cardinality estimators, but may flip coins during computation. We denote by $\mathfrak{M}$ the set of distributions over $\mathcal{M}$.

Definition 8. A *probabilistic cardinality estimator* is a tuple $\langle M_\varnothing, add, merge, estimate \rangle$, where
– $M_\varnothing \in \mathcal{M}$ is the *empty sketch*;
– $add\,(M, e) : \mathcal{M} \times \mathcal{U} \longrightarrow \mathfrak{M}$ is the probabilititistic operation that adds the element e to the sketch M and returns an updated sketch;
– $merge\,(M_1, M_2) : \mathcal{M} \times \mathcal{M} \longrightarrow \mathfrak{M}$ is the probabilistic operation that merges two sketches M_1 and M_2; and
– $estimate\,(M) : \mathcal{M} \longrightarrow \mathbb{N}$ estimates the number of unique elements that have been added to the sketch.

Both the *add* and *merge* operations can be extended to *distributions* of sketches. For a distribution of sketches $\mathcal{D}$ and an element e, $add\,(\mathcal{D}, e)$ denotes the distribution such that:

$$\mathbb{P}\left[add\,(\mathcal{D}, e) = M_0\right] = \sum_M \mathbb{P}\left[\mathcal{D} = M\right] \mathbb{P}\left[add\,(M, e) = M_0\right].$$

For two distributions of sketches $\mathcal{D}$ and $\mathcal{D}'$, $merge\,(\mathcal{D}, \mathcal{D}')$ denotes the distribution such that:

$$\mathbb{P}\left[merge\,(\mathcal{D}, \mathcal{D}') = M_0\right]$$
$$= \sum_{M, M'} \mathbb{P}\left[\mathcal{D} = M\right] \mathbb{P}\left[\mathcal{D}' = M'\right] \mathbb{P}\left[merge\,(M, M') = M_0\right].$$

We want probabilistic cardinality estimators to have the same high-level properties as deterministic cardinality estimators: idempotence, commutativity, and the existence of a well-behaved *merge* operation. In the deterministic case, the idempotence and commutativity of the *add* operation was sufficient to show the existence of a *merge* operation with the desired properties. In the probabilistic case, this no longer holds. Instead, we require the following two properties.

– For a set $E \subseteq \mathcal{U}$, let $\mathcal{D}_E$ denote the sketch distribution obtained when adding elements of E successively into $M_\varnothing$. The mapping from E to $\mathcal{D}_E$ must be well-defined: it must be independent of the order in which we add elements, and possible repetitions. This requirement encompasses both idempotence and commutativity.
– For two subsets E_1 and E_2 of $\mathcal{U}$, we require that $merge\,(\mathcal{D}_{E_1}, \mathcal{D}_{E_2}) = \mathcal{D}_{E_1 \cup E_2}$.

These requirements encompass the results of Lemma 5.

These properties, like in the deterministic case, are very strong. They impose that an arbitrary number of sketches can be aggregated without losing accuracy during the aggregation process. This requirement is however realistic in many practical contexts, where the same sketches are used for fine-grained analysis and for large-scale cardinality estimation. If this requirement is relaxed, and the cardinality estimator is allowed to return imprecise results when merging sketches, our negative results do not hold.

For example, Tschorsch and Scheuermann proposed a cardinality estimation scheme [41] which adds noise to sketches to make them satisfy privacy guarantees in a distributed context. However, their algorithm is not a probabilistic cardinality estimator according to our definition: noisy sketches can no longer be aggregated. Indeed, [41] explains that "combining many perturbed sketches quickly drives [noise] to exceedingly high values." In our setting, aggregation is crucial, so we do not further consider their algorithm.

4 Modeling privacy

In this section, we describe our system and attacker model (4.1), and present our privacy definition (4.2).

4.1 System and attacker model

Figure 1 shows our system and attacker model. The service provider collects sensitive data from many users over a long time span. The raw data is stored in a database. Over shorter time periods (e.g. an hour, a day, or a week), a cardinality estimator aggregates all data into a sketch. Sketches of all time periods are stored in a sketch database. Sketches from different times are aggregated into sketches of longer time spans, which are also stored in the database. Estimators compute user
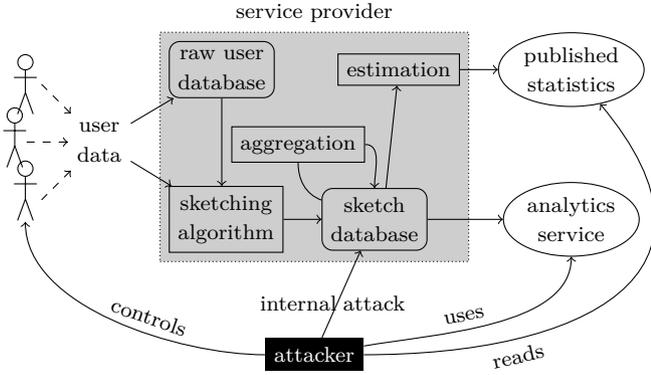
Fig. 1. System and attacker model

statistics from the sketches in the database, which are published. The service provider may also publish the sketches via an analytics service for other parties.

The attacker knows all algorithms used (those for sketching, aggregation, and estimation, including their configuration parameters such as the hash function and the number of buckets) and has access to the published statistics and the analytics service. She controls a small fraction of the users that produce user data. However, she can neither observe nor change the data of the other users. She also does not have access to the database containing the raw data.

In this work, we mainly consider an *internal attacker* who has access to the sketch database. For this internal attacker, the goal is to discover whether her target belongs to a given sketch. We then discuss how our results extend to weaker *external attackers*, which can only use the analytics service. We will see that for our main results, the attacker only requires access to one sketch. The possibility to use multiple sketches will only come up when discussing mitigations strategies in Section 8.1.

4.2 Privacy definition

We now present the privacy definition used in our main result. Given the system and attacker just described, our definition captures the impossibility for the attacker to gain significant positive knowledge about a given target. We explain this notion of knowledge gain, state assumptions on the attacker's prior knowledge, and compare our privacy notion with other well-known definitions.

We define a very *weak* privacy requirement: reasonable definitions used for practical algorithms would likely be stronger. Working with a weak definition *strengthens* our negative result: if a cardinality estima-

tor satisfying our weak privacy definition cannot be precise, then this is also the case for cardinality estimators satisfying a stronger definition.

In Section 6, we explore even weaker privacy definitions and prove similar negative results (although with a looser bound). In Section 7, we relax the requirement that *every individual user* must be protected according to the privacy definition. We show then that our theorem no longer holds, but that practical uses of cardinality estimators still cannot be considered privacy-preserving.

We model a possible attack as follows. The attacker has access to the identifier of a user t (her *target*) and a sketch M generated from a set of users E ($M \leftarrow M_E$) unknown to her. The attacker wants to know whether $t \in E$. She initially has a *prior* knowledge of whether the target is in the sketch. Like in Bayesian inference, $\frac{\mathbb{P}[t \in E]}{\mathbb{P}[t \notin E]}$ represents how much more likely the user is in the database than is not, according to the attacker. After looking at the sketch M, this knowledge changes: her *posterior* knowledge becomes $\frac{\mathbb{P}[t \in E \mid M_E = M]}{\mathbb{P}[t \notin E \mid M_E = M]}$.

We define privacy to capture that the attacker's posterior knowledge should not increase too much. In other words, the attacker should not gain significant knowledge by seeing the sketch. This must hold for every possible sketch M and every possible user t. We show in Lemma 10 that the following definition bounds the *positive* knowledge gain of the attacker.

Definition 9. A cardinality estimator satisfies ε-*sketch privacy above cardinality* N if for every $n \geq N$, $t \in \mathcal{U}$, and $M \in \mathcal{M}$, the following inequality holds:

$$\mathbb{P}_n \left[M_E = M \mid t \in E \right] \leq e^{\varepsilon} \cdot \mathbb{P}_n \left[M_E = M \mid t \notin E \right].$$

Here, the probability $\mathbb{P}_n$ is taken over:
- a uniformly chosen set $E \in \mathcal{P}_n \left(\mathcal{U} \right)$, where $\mathcal{P}_n \left(\mathcal{U} \right)$ is the set of all possible subsets $E \subseteq \mathcal{U}$ of cardinality k; and
- the coin flips of the algorithm, for probabilistic cardinality estimators.

If a cardinality estimator satisfies this definition, then for any user t, the probability of observing M if $t \in E$ is not much higher than the probability of observing M if $t \notin E$. To give additional intuition on Definition 9, we now show that the parameter ε effectively captures the attacker's positive knowledge gain.

Lemma 10. *A cardinality estimator satisfies ε-sketch privacy above cardinality N if and only if the following inequality holds for every $n \geq N$, $t \in \mathcal{U}$ and $M \in \mathcal{M}$*

with $\mathbb{P}_n [M_E = M] > 0$:

$$\frac{\mathbb{P}_n [t \in E \mid M_E = M]}{\mathbb{P}_n [t \notin E \mid M_E = M]} \le e^\varepsilon \cdot \frac{\mathbb{P}_n [t \in E]}{\mathbb{P}_n [t \notin E]}.$$

Proof. Bayes' law can be used to derive one inequality from the other. We have

$$\mathbb{P}_n[M_E = M \mid t \in E] = \frac{\mathbb{P}_n[t \in E \mid M_E = M] \cdot \mathbb{P}_n[M_E = M]}{\mathbb{P}_n[t \in E]}$$

$$\mathbb{P}_n[M_E = M \mid t \notin E] = \frac{\mathbb{P}_n[t \notin E \mid M_E = M] \cdot \mathbb{P}_n[M_E = M]}{\mathbb{P}_n[t \notin E]}.$$

The equivalence between the definitions follows directly. ☐

This definition has three characteristics which make it unusually weak. They correspond to an *under-approximation* of the attacker's capabilities and goals.

Uniform prior The choice of distribution for $\mathbb{P}_n$ implies that the elements of E are *uniformly distributed* in $\mathcal{U}$. This corresponds to an attacker who has *no prior knowledge* about the data. In the absence of prior information about the elements of the set E, the attacker's best approximation is the uniform distribution. In practice, a realistic attacker might have more information about the data, so a stronger privacy definition would model this prior knowledge by a larger family of probability distributions. More precisely, since the elements of E are uniformly distributed in $\mathcal{U}$, the prior knowledge from the attacker $\mathbb{P}_n [t \in E]$ is exactly $|E| / |\mathcal{U}|$. A realistic attacker would likely have a larger prior knowledge about their target. However, any reasonable definition of privacy would also include the case where the attacker does not have more information on their target than on other users and, as such, would be stronger than ϵ-sketch privacy.

Asymmetry We only consider the *positive* information gain by the attacker. There is an *upper bound* on the probability that $t \in E$ given the observation M, but no lower bound. In other words, the attacker is allowed to deduce with absolute certainty that $t \notin E$. In practice, both positive *and* negative information gains may present a privacy risk. In our running example (see Example 1), deducing that a user did *not* spend the night at his apartment could be problematic.

Minimum cardinality We only require a bound on the information gain for cardinalities larger than a parameter N. In practice, N could represent a threshold over which it is considered safe to publish sketches or to relax data protection requirements. Choosing a small N (like $N = 10$) strengthens the privacy definition, while

choosing a large N (like $N = 500$) limits the utility of the data, as many smaller sketches cannot be published.

We emphasize again that these characteristics, which result in a very weak definition, make our notion of privacy well-suited to proving *negative results*. If satisfying our definition is impossible for an accurate cardinality estimator, then a stronger definition would similarly be impossible to satisfy. For example, any reasonable choice of distributions used to represent the prior knowledge of the attacker would include the uniform distribution.

We now compare our definition to two other notions: differential privacy [17] and Pufferfish privacy [30].

4.3 Relation to differential privacy

Recall the definition of differential privacy: $\mathcal{A}$ is ε-differentially private if and only if $e^{-\varepsilon} \le \frac{\mathbb{P}[\mathcal{A}(D_1)]}{\mathbb{P}[\mathcal{A}(D_2)]} \le e^\varepsilon$ for any databases D_1 and D_2 that only differ by one element. In our setup, this could be written as the two inequalities $\mathbb{P}[M_E \mid t \in E] \le e^\varepsilon \cdot \mathbb{P}[M_E \mid t \notin E]$ and $\mathbb{P}[M_E \mid t \notin E] \le e^\varepsilon \cdot \mathbb{P}[M_E \mid t \in E]$.

Asymmetry, and minimum cardinality, are two obvious differences between our notion of privacy and differential privacy. But the major difference lies in the *source of uncertainty*. In differential privacy, the probabilities are taken over the coin flips of the algorithm. The attacker is implicitly assumed to know the algorithm's input except for *one* user: the uncertainty comes entirely from the algorithm's randomness. In our definition, the attacker has no prior knowledge of the input, so the uncertainty comes either entirely from the attacker's lack of background knowledge (for deterministic cardinality estimators), or both from the attacker's lack of background knowledge and the algorithm's inherent randomness.

The notion of relying on the initial lack of knowledge of the attacker in a privacy definition is not new: it is for example made explicit in the definition of Pufferfish privacy, a generic framework for privacy definitions.

4.4 Relation to Pufferfish privacy

Pufferfish privacy [30] is a customizable framework for building privacy definitions. A Pufferfish privacy definition has three components: a set of *potential secrets* $\mathbb{S}$; a set of *discriminative pairs* $\mathbb{S}_{\text{pairs}} \subseteq \mathbb{S} \times \mathbb{S}$; and a set of *data evolution scenarios* $\mathbb{D}$.

$\mathbb{S}_{\text{pairs}}$ represents the facts we want the attacker to be unable to distinguish. In our case, we want to prevent the attacker from distinguishing between $t \in E$ and $t \notin E$: $\mathbb{S}_{\text{pairs}} = \{(t \in E, t \notin E) \mid t \in \mathcal{U}\}$. $\mathbb{D}$ represents what the possible distributions of the input data are. In our case, it is a singleton that only contains the uniform distribution.

Our definition is *almost* an instance of Pufferfish privacy. Like with differential privacy, the main difference is asymmetry.

The close link to Pufferfish privacy supports our proof of two fundamental properties of privacy definitions: *transformation invariance* and *convexity* [30]. Transformation invariance states that performing additional analysis of the output of the algorithm does not allow an attacker to gain more information, i.e., the privacy definition is closed under composition with probabilistic algorithms. Convexity states that if a data owner chooses randomly between two algorithms satisfying a privacy definition and generates the corresponding output, this procedure itself will satisfy the same privacy definition. These two properties act as *sanity checks* for our privacy definition.

Proposition 11. *ε-sketch privacy above cardinality N satisfies transformation invariance and convexity.*

Proof. The proof is similar to the proof of Theorem 5.1 in [30], proved in Appendix B of the same paper. □

5 Private cardinality estimators are imprecise

Let us return to our privacy problem: someone with access to a sketch wants to know whether a given individual belongs to the aggregated individuals in the sketch. Formally, given a target t and a sketch M_E, the attacker must guess whether $t \in E$ with high probability. In Section 5.1, we explain how the attacker can use a simple test to gain significant information if the cardinality estimator is deterministic. Then, in Section 5.2, we reformulate the main technical lemma in probabilistic terms, and prove an equivalent theorem for probabilistic cardinality estimators.

5.1 Deterministic case

Given a target t and a sketch M_E, the attacker can perform the following simple attack to guess whether $t \in E$. She can try to add the target t to the sketch M_E, and observe whether the sketch changes. In other words, she checks whether $add(M_E, t) = M_E$. If the sketch changes, this means with certainty that $t \notin E$. Thus, Bayes' law indicates that if $add(M_E, t) = M_E$, then the probability of $t \in E$ cannot decrease.

How large is this increase? Intuitively, it depends on how likely it is that adding an element to a sketch does not change it *if the element has not previously been added to the sketch*. Formally, it depends on $\mathbb{P}[add(M_E, t) = M_E \mid t \notin E]$.

– If $\mathbb{P}[add(M_E, t) = M_E \mid t \notin E]$ is close to 0, for example if the sketch is a list of all elements seen so far, then observing that $add(M_E, t) = M_E$ will lead the attacker to believe with high probability that $t \in E$.
– If $\mathbb{P}[add(M_E, t) = M_E \mid t \notin E]$ is close to 1, it means that adding an element to a sketch often does not change it. The previous attack does not reveal much information. But then, it also means that many elements are ignored when they are added to the sketch, that is, the sketch does not change when adding the element. Intuitively, the accuracy of an estimator based solely on a sketch that ignores many elements cannot be very good.

We formalize this intuition in the following theorem.

Theorem 12. *An unbiased deterministic cardinality estimator that satisfies ε-sketch privacy above cardinality N is not precise. Namely, its variance is at least $\frac{1 - c^k}{c^k}(n - k \cdot N)$, for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - e^{-\varepsilon}$*

Note that if we were using differential privacy, this result would be trivial: no deterministic algorithm can ever be differentially private. However, this is not so obvious for our definition of privacy: prior work [7, 11, 28] shows that when the attacker is assumed to have some uncertainty about the data, even deterministic algorithms can satisfy the corresponding definition of privacy.

Figure 2 shows plots of the lower bound on the standard error of a cardinality estimator with ε-sketch privacy at two cardinalities (100 and 500). It shows that the standard error increases exponentially with the number of elements added to the sketch. This demonstrates that even if we require the privacy property for a large
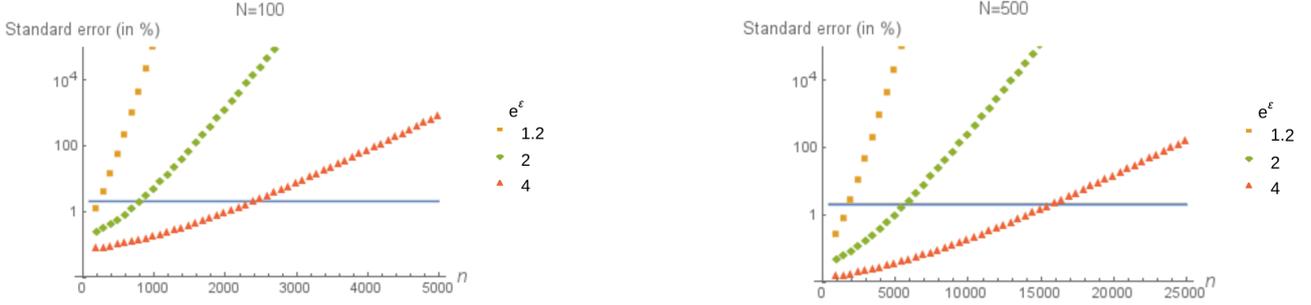
Fig. 2. Minimum standard error for a cardinality estimator with ε-sketch privacy above cardinality 100 (left) and 500 (right). The blue line is the relative standard error of HyperLogLog with standard parameters.

value of N (500) and a large ε (which is generally less than 1), the standard error of a cardinality estimator will become unreasonably large after 20,000 elements.

Proof of Theorem 12. The proof is comprised of three steps, following the intuition previously given.

1. We show that a sketch M_E, computed from a random set E with an ε-sketch private estimator above cardinality N, will ignore many elements after N (Lemma 13).
2. We prove that if a cardinality estimator ignores a certain ratio of elements after adding $n = N$ elements, then it will ignore an even larger ratio of elements as n increases (Lemma 14).
3. We conclude by proving that an unbiased cardinality estimator that ignores many elements must have a large variance (Lemma 15).

The theorem follows directly from these lemmas. $\qquad\square$

Lemma 13. *Let $t \in \mathcal{U}$. A deterministic cardinality estimator with ε-sketch privacy above cardinality N satisfies $\mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \notin E \right] \geq e^{-\varepsilon}$ for $n \geq N$.*

Proof. We first prove that such an estimator satisfies

$$\mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \in E \right]$$
$$\leq e^{\varepsilon} \cdot \mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \notin E \right].$$

We decompose the left-hand side of the inequality over all possible values of M_E which that $add \left(M_E, t \right) = M_E$. If we call this set $\mathcal{I}_t = \{ M \mid add \left(M, t \right) = M \}$, we have:

$$\mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \in E \right]$$
$$= \sum_{M \in \mathcal{I}_t} \mathbb{P}_n \left[M_E = M \mid t \in E \right]$$
$$\leq e^{\varepsilon} \cdot \sum_{M \in \mathcal{I}_t} \mathbb{P}_n \left[M_E = M \mid t \notin E \right]$$
$$\leq e^{\varepsilon} \cdot \mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \notin E \right],$$

where the first inequality is obtained directly from the definition of ε-sketch privacy.

Now, Lemma 3 gives $\mathbb{P} \left[add \left(M_E, t \right) = M_E \mid t \in E \right] = 1$, and finally $\mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \notin E \right] \geq e^{-\varepsilon}$. $\quad\square$

Lemma 14. *Let $t \in \mathcal{U}$. Suppose a deterministic cardinality estimator satisfies $\mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \notin E \right] \geq p$ for any $n \geq N$. Then for any integer $k \geq 1$, it also satisfies $\mathbb{P}_n \left[add \left(M_E, t \right) = M_E \mid t \notin E \right] \geq 1 - (1-p)^k$, for $n \geq k \cdot N$.*

Proof. First, note that if $F \subseteq E$, and $add \left(M_F, t \right) = M_F$, then $add \left(M_E, t \right) = M_E$. This is a direct consequence of Lemma 5: $M_E = merge \left(M_{E \setminus F}, M_F \right)$, so:

$$add \left(M_E, t \right) = merge \left(M_{E \setminus F}, add \left(M_F, t \right) \right)$$
$$= merge \left(M_{E \setminus F}, M_F \right)$$
$$= M_E$$

We show next that when $n \geq k \cdot N$, generating a set $E \in \mathcal{P}_n \left(\mathcal{U} \right)$ uniformly randomly can be seen as generating k *independent* sets in $\mathcal{P}_N \left(\mathcal{U} \right)$, then merging them. Indeed, generating such a set can be done by as follows:

1. For $i \in \{1, \ldots, k\}$, generate a set $E_i \subseteq \mathcal{P}_N \left(\mathcal{U} \right)$ uniformly randomly. Let $E_{\cup} = \bigcup_i E_i$.
2. Count the number of elements appearing in multiple E_i: $d = \# \{ x \in E_i \mid \exists j < i, x \in E_j \}$. Generate a set $E' \in \mathcal{P}_{n-d} \left(\mathcal{U} \setminus E_{\cup} \right)$ uniformly randomly.

E is then defined by $E = E_{\cup} \cup E'$. Step 1 ensures that we used k *independent* sets of cardinality N to generate E, and step 2 ensures that E has exactly n elements.

Intuitively, each time we generate a set E_i of cardinality N uniformly at random in $\mathcal{U}$, we have *one chance* that t will be ignored by E_i (and thus by E). So t can be ignored by M_E with a certain probability because it was ignored by M_{E_1}. Similarly, it can also be ignored be-

cause of M_{E_2}, etc. Since the choice of E_i is independent of the choice of elements in $\bigcup_{j\neq i} E_j$, we can rewrite:

$$\mathbb{P}_n\left[add\left(M_E, t\right) \neq M_E \mid t \notin E\right]$$
$$\leq \prod_{i=1}^{k} \mathbb{P}_n\left[add\left(M_{E_i^0}, t\right) \neq M_{E_i^0} \mid t \notin E\right]$$
$$\leq \prod_{i=1}^{k}\left(1 - \mathbb{P}_n\left[add\left(M_{E_i^0}, t\right) = M_{E_i^0} \mid t \notin E_i\right]\right)$$
$$\leq (1-p)^k$$

using the hypothesis of the lemma. Thus:

$$\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \geq 1 - (1-p)^k.$$

$\square$

Lemma 15. *Suppose a deterministic cardinality estimator satisfies $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \geq 1-p$ for any $n \geq N$ and all t. Then its variance for $n \geq N$ is at least $\frac{1-p}{p}(n - N)$.*

Proof. The proof's intuition is as follows. The hypothesis of the lemma requires that the cardinality estimator, on average, *ignores* a proportion $1 - p$ of new elements added to a sketch (once N elements have been added): the sketch is not changed when a new element is added. The best thing that the cardinality estimator can do, then, is to store all elements that it does not ignore, count the number of unique elements among these, and multiply this number by $1/p$ to correct for the elements ignored. It is well-known that estimating the size k of a set based on the size of a uniform sample of sampling ratio p has a variance of $\frac{1-p}{p}k$. Hence, our cardinality estimator has a variance of at least $\frac{1-p}{p}(n - N)$.

Formalizing this idea requires some additional technical steps. The full proof is given in Appendix B. $\square$

All existing cardinality estimators satisfy our axioms and their standard error remains low even for large values of n. Theorem 12 shows, for all of them, that there are some users whose privacy loss is significant. In Section 7.2, we quantify this precisely for HyperLogLog.

5.2 Probabilistic case

Algorithms that add noise to their output, or more generally, are allowed to use a source of randomness, are often used in privacy contexts. As such, even though all cardinality estimators used in practical applications are deterministic, it is reasonable to hope that a probabilistic cardinality estimator could satisfy our very weak privacy definition. Unfortunately, this is not the case.

In the deterministic case, we showed that for any element t, the probability that t has an influence on a random sketch M decreases exponentially with the sketch size. Or, equivalently, the distribution of sketches of size kn that do not contain t is "almost the same" (up to a density of probability $\left(1 - e^{-\varepsilon}\right)^k$) as the distribution of sketches of the same size, but containing t.

The following lemma establishes the same result in the probabilistic setting. Instead of reasoning about the probability that an element t is "ignored" by a sketch M, we reason about the probability that t has a *meaningful* influence on this sketch. We show that this probability decreases exponentially, even if $\mathbb{P}\left[M \neq add\left(M, t\right)\right]$ is very high.

First, we prove a technical lemma on the *structure* that the *merge* operation imposes on the space of sketch distributions. Then, we find an upper bound on the "meaningful influence" of an element t, when added to a random sketch of cardinality n. We then use this upper bound, characterized using the statistical distance, to show that the estimator variance is as imprecise as for the deterministic case.

Definition 16. Let $\mathbb{D}$ be the real vector space spanned by the family $\{\mathcal{D}_E \mid E \subseteq \mathcal{U}\}$ (seen as vectors of $\mathbb{R}^{\mathcal{M}}$). For any probability distributions $\mathcal{A}, \mathcal{B} \in \mathbb{D}$, we denote $\mathcal{A} \cdot \mathcal{B} = merge\left(\mathcal{A}, \mathcal{B}\right)$. We show in Lemma 17 that this notation makes sense: on $\mathbb{D}$, we can do computations as if *merge* was a multiplicative operation.

Lemma 17. *The merge operation defines a commutative and associative algebra on $\mathbb{D}$.*

Proof. By the properties required from probabilistic cardinality estimators in Definition 8, the *merge* operation is commutative and associative on the family $\{\mathcal{D}_E \mid E \subseteq \mathcal{U}\}$. By linearity of the *merge* operation, these properties are preserved for any linear combination of vectors $\mathcal{D}_E$. $\square$

Lemma 18. *Suppose a cardinality estimator satisfies ε-sketch privacy above cardinality N, and let $t \in \mathcal{U}$. Let $\mathcal{D}_{out,n}$ be the distribution of sketches obtained by adding n uniformly random elements of $\mathcal{U} \setminus \{t\}$ into $M_\varnothing$ (or, equivalently, $\mathcal{D}_{out,n}(M) = \mathbb{P}_n\left[M_E = M \mid t \notin E\right]$). Then:*

$$\upsilon\left(\mathcal{D}_{out,kn}, add\left(\mathcal{D}_{out,kn}, t\right)\right) \leq \left(1 - e^{-\varepsilon}\right)^k$$

where υ is the statistical distance between probability distributions.

Proof. Let $\mathcal{D}_{\mathrm{in},n}$ be the distribution of sketches obtained by adding t, then $n-1$ uniformly random elements of $\mathcal{U}$ into M (or, equivalently, $\mathcal{D}_{\mathrm{in},n}(M) = \mathbb{P}_n[M_E = M | t \in E]$). Then the definition of ε-sketch privacy gives that for every sketch M, $\mathcal{D}_{\mathrm{out},n}(M) \geq e^{-\varepsilon}\mathcal{D}_{\mathrm{in},n}(M)$. So we can express $\mathcal{D}_{\mathrm{out},n}$ as the *sum* of two distributions:

$$\mathcal{D}_{\mathrm{out},n} = e^{-\varepsilon}\mathcal{D}_{\mathrm{in},n} + \left(1 - e^{-\varepsilon}\right)\mathcal{R}$$

for a certain distribution $\mathcal{R}$.

First, we show that $\mathcal{D}_{\mathrm{out},kn} = (\mathcal{D}_{\mathrm{out},n})^k \cdot \mathcal{C}$ for a certain distribution $\mathcal{C}$. Indeed, to generate a sketch of cardinality kn that does not contain t uniformly randomly, one can use the following process.

1. Generate k random sketches of cardinality n which do not contain t, and merge them.
2. For all $E \subseteq \mathcal{U}$, denote by p_E the probability that the k sketches were generated with the elements in E. There might be "collisions" between the k sketches: if several sketches were generated using the same element, $|E| < kn$. When this happens, we need to "correct" the distribution, and add additional elements. Enumerating all the options, we denote $\mathcal{C} = \sum p_E \mathcal{D}^{\mathrm{c}}_{E,nk}$, where $\mathcal{D}^{\mathrm{c}}_{E,nk}$ is obtained by adding $nk - |E|$ uniformly random elements in $\mathcal{U} \setminus E$ to $\mathcal{M}_\varnothing$. Thus, $\mathcal{D}_{\mathrm{out},kn} = (\mathcal{D}_{\mathrm{out},n})^k \cdot \mathcal{C}$.

All these distributions are in $\mathbb{D}$: $\mathcal{D}_{\mathrm{out},n} = \mathrm{avg}_{E \in \mathcal{P}_n(\mathcal{U}), t \notin E}\mathcal{D}_E$, $\mathcal{D}_{\mathrm{in},n} = \mathrm{avg}_{E \in \mathcal{P}_n(\mathcal{U}), t \in E}\mathcal{D}_E$, $\mathcal{R} = \left(1 - e^{-\varepsilon}\right)^{-1}\left(\mathcal{D}_{\mathrm{out},n} - e^{-\varepsilon}\mathcal{D}_{\mathrm{in},n}\right)$, etc. Thus:

$$\mathcal{D}_{\mathrm{out},kn} = (\mathcal{D}_{\mathrm{out},n})^k \cdot \mathcal{C}$$
$$= \left(e^{-\varepsilon}\mathcal{D}_{\mathrm{in},n} + \left(1 - e^{-\varepsilon}\right)\mathcal{R}\right)^k \cdot \mathcal{C}$$
$$= \sum_{i=0}^{k}\binom{k}{i}e^{-i\cdot\varepsilon}\left(1 - e^{-\varepsilon}\right)^{k-i}\mathcal{D}^i_{\mathrm{in},n} \cdot \mathcal{R}^{k-i} \cdot \mathcal{C}.$$

Denoting $\mathcal{A} = \sum_{i=1}^{k}\binom{k}{i}e^{-i\cdot\varepsilon}\left(1 - e^{-\varepsilon}\right)^{k-i}\mathcal{D}^{i-1}_{\mathrm{in},n} \cdot \mathcal{R}^{k-i} \cdot \mathcal{C}$ and $\mathcal{B} = \mathcal{R}^k \cdot \mathcal{C}$, this gives us:

$$\mathcal{D}_{\mathrm{out},kn} = \mathcal{A} \cdot \mathcal{D}_{\mathrm{in},n} + \left(1 - e^{-\varepsilon}\right)^k\mathcal{B}.$$

Finally, we can compute $add\left(\mathcal{D}_{\mathrm{out},kn}, t\right)$:

$$add\left(\mathcal{D}_{\mathrm{out},kn}, t\right) = \mathcal{A} \cdot \mathcal{D}_{\mathrm{in},n} \cdot \mathcal{D}_{\{t\}} + \left(1 - e^{-\varepsilon}\right)^k\mathcal{B} \cdot \mathcal{D}_{\{t\}}$$
$$= \mathcal{A} \cdot \mathcal{D}_{\mathrm{in},n} + \left(1 - e^{-\varepsilon}\right)^k\mathcal{B} \cdot \mathcal{D}_{\{t\}}$$

Note that since $\mathcal{D}_{\mathrm{in},n} = \mathrm{avg}_{E \in \mathcal{P}_n(\mathcal{U}), t \in E}\mathcal{D}_E$, we have $\mathcal{D}_{\mathrm{in},n} \cdot \mathcal{D}_{\{t\}} = \mathcal{D}_{\mathrm{in},n}$ by idempotence, and:

$$\upsilon\left(\mathcal{D}_{\mathrm{out},kn}, add\left(\mathcal{D}_{\mathrm{out},kn}, t\right)\right)$$
$$= \frac{1}{2}\left\|\mathcal{D}_{\mathrm{out},kn} - add\left(\mathcal{D}_{\mathrm{out},kn}, t\right)\right\|_1$$
$$= \frac{1}{2}\left\|\left(1 - e^{-\varepsilon}\right)^k\mathcal{B} - \left(1 - e^{-\varepsilon}\right)^k\mathcal{B} \cdot \mathcal{D}_{\{t\}}\right\|_1$$
$$\leq \frac{\left(1 - e^{-\varepsilon}\right)^k}{2}\left(\|\mathcal{B}\|_1 + \left\|\mathcal{B} \cdot \mathcal{D}_{\{t\}}\right\|_1\right)$$
$$\leq \left(1 - e^{-\varepsilon}\right)^k.$$

$\square$

Lemma 18 is the probabilistic equivalent of Lemmas 13 and 14. Now, we state the equivalent of Lemma 15, and explain why its intuition still holds in the probabilistic case.

Lemma 19. *Suppose that a cardinality estimator satisfies for any $n \geq N$ and all t, $\upsilon\left(\mathcal{D}_{out,n}, add\left(\mathcal{D}_{out,n}, t\right)\right) \leq p$. Then its variance for $n \geq N$ is at least $\frac{1-p}{p}(n - N)$.*

Proof. The condition "$\upsilon\left(\mathcal{D}_{out,n}, add\left(\mathcal{D}_{out,n}, t\right)\right) \leq p$" is equivalent to the condition of Lemma 15: with probability $(1-p)$, the cardinality estimator "ignores" when a new element t is added to a sketch. Just like in Lemma 15's proof, we can convert this constraint into estimating the size of a set based on a sampling set. The best known estimator for this problem is deterministic, so allowing the cardinality estimator to be probabilistic does not help improving the optimal variance.

The same result than in Lemma 15 follows. $\square$

Lemmas 18 and 19 together immediately lead to the equivalent of Theorem 12 in the probabilistic case.

Theorem 20. *An unbiased probabilistic cardinality estimator that satisfies ε-sketch privacy above cardinality N is not precise. Namely, its variance is at least $\frac{1-c^k}{c^k}(n - k \cdot N)$, for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - e^{-\varepsilon}$*

Somewhat surprisingly, allowing the algorithm to add noise to the data seems to be pointless from a privacy perspective. Indeed, given the same privacy guarantee, the lower bound on accuracy is the same for deterministic and probabilistic cardinality estimators. This suggests that the constraints of these algorithms (idempotence and commutativity) require them to somehow keep a trace of who was added to the sketch (at least for some users), which is fundamentally incompatible with even weak notions of privacy.

6 Weakening the privacy definition

Our main result is negative: no cardinality estimator satisfying our privacy definition can maintain a good accuracy. Thus, it is natural to wonder whether our privacy definition is too strict, and if the result still holds for weaker variants.

In this section, we consider two weaker variants of our privacy definition: one allows a small probability of privacy loss, while the other averages the privacy loss across all possible outputs. We show that these natural relaxations do not help as close variants of our negative result still hold.

6.1 Allowing a small probability of privacy loss

As Lemma 10 shows, ε-sketch differential privacy provides a bound on how much information the attacker can gain in the worst case. A natural relaxation is to accept a small *probability of failure*: requiring a bound on the information gain in *most cases*, and accept a potentially unbounded information gain with low probability.

We introduce a new parameter, called δ, similar to the use of δ in the definition of (ε, δ)-differential privacy: $\mathcal{A}$ is (ε, δ)-differentially private if and only if for any databases D_1 and D_2 that only differ by one element and any set S of possible outputs, $\mathbb{P}\left[\mathcal{A}\left(D_1\right) \in S\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{A}\left(D_2\right) \in S\right] + \delta$.

Definition 21. A cardinality estimator satisfies (ε, δ)-*sketch privacy above cardinality N* if for every $\mathcal{S} \subseteq \mathcal{M}$, $n \geq N$, and $t \in \mathcal{U}$,

$$\mathbb{P}_n\left[M_E \in \mathcal{S} \mid t \in E\right] \leq e^{\varepsilon} \cdot \mathbb{P}_n\left[M_E \in \mathcal{S} \mid t \notin E\right] + \delta.$$

Unfortunately, our negative result still holds for this variant of the definition. Indeed, we show that a close variant of Lemma 13 holds, and the rest follows directly.

Lemma 22. *Let $t \in \mathcal{U}$. A cardinality estimator that satisfies (ε, δ)-probabilistic sketch privacy above cardinality N satisfies $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \geq \left(\frac{1}{2} - \delta\right) \cdot e^{-\varepsilon}$ for $n \geq N$.*

The proof of Lemma 22 is given in Appendix C. We can then deduce a theorem similar to our negative result for our weaker privacy definition.

Theorem 23. *An unbiased cardinality estimator that satisfies (ε, δ)-sketch privacy above cardinality N has a variance at least $\frac{1-c^k}{c^k}\left(n - k \cdot N\right)$ for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - \left(\frac{1}{2} - \delta\right) \cdot e^{-\varepsilon}$. It is therefore not precise if $\delta < \frac{1}{2}$.*

Proof. This follows from Lemmas 22, 14 and 15. $\quad\square$

6.2 Averaging the privacy loss

Instead of requiring that the attacker's information gain is bounded by ε for every possible output, we could bound the *average* information gain. This is equivalent to accepting a larger privacy loss in some cases, as long as other cases have a lower privacy loss.

This intuition is captured by the use of Kullback-Leibler divergence, which is often used in similar contexts [14, 19, 38, 39]. In our case, we adapt it to maintain the asymmetry of our original privacy definition. First, we give a formal definition the *privacy loss* of a user t given output M.

Definition 24. Given a cardinality estimator, the *positive privacy loss of t given output M at cardinality n* is defined as

$$\varepsilon_{n,t,M} = \max\left(\log\left(\frac{\mathbb{P}_n\left[M_E = M \mid t \in E\right]}{\mathbb{P}_n\left[M_E = M \mid t \notin E\right]}\right), 0\right).$$

This privacy loss is never negative: this is equivalent to discarding the case where the attacker gains *negative* information. Now, we bound this average over all possible values of M_E, given $t \in E$.

Definition 25. A cardinality estimator satisfies ε-*sketch average privacy above cardinality N* if for every $n \geq N$ and $t \in \mathcal{U}$, we have

$$\sum_M \mathbb{P}_n\left[M_E = M \mid t \in E\right] \cdot \varepsilon_{n,t,M} \leq \varepsilon.$$

It is easy to check that ε-sketch average privacy above cardinality N is strictly weaker than ε-sketch privacy above cardinality N. Unfortunately, this definition is also stronger than $(\varepsilon_\delta, \delta)$-sketch privacy above cardinality N for certain values of ε and δ, and as such, Lemma 22 also applies. We prove this in the following lemma.

Lemma 26. *If a cardinality estimator satisfies ε-sketch average privacy above cardinality N, then it also satisfies $\left(\frac{\varepsilon}{\delta}, \delta\right)$-sketch privacy above cardinality N for any $\delta > 0$.*

The proof is given in Appendix D. This lemma leads to a similar version of the negative result.

Theorem 27. *An unbiased cardinality estimator that satisfies ε-sketch average privacy above cardinality N has a variance at least $\frac{1-c^k}{c^k}(n - k \cdot N)$ for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - \frac{e^{-4\varepsilon}}{4}$. It is thus not precise.*

Proof. This follows directly from Lemma 26 with $\delta = \frac{1}{4}$, and Theorem 23. □

Recall that all existing cardinality estimators satisfy our axioms and have a bounded accuracy. Thus, an immediate corollary is that for all cardinality estimators used in practice, there are some users for which the *average* privacy loss is very large.

Remark 28. This idea of *averaging ε* is similar to the idea behind Rényi differential privacy [33]. The parameter α of Rényi differential privacy determines the averaging method used (geometric mean, arithmetic mean, quadratic mean, etc.). Using KL-divergence corresponds to $\alpha = 1$, while $\alpha = 2$ averages all possible values of e^ε. Increasing α strengthens the privacy definition [33, Prop. 9], so our negative result still holds.

7 Privacy loss of individual users

So far, we only considered definitions of privacy that give the same guarantees for all users. What if we allow certain users to have less privacy than others, or if we were to average the privacy loss *across users* instead of averaging over all possible outcomes for each user?

Such definitions would generally not be sufficiently convincing to be used in practice: one typically wants to protect *all users*, not just a majority of them. In this section, we show that even if we relax this requirement, cardinality estimators would in practice leak a significant amount of information.

7.1 Allowing unbounded privacy loss for some users

What happens if we allow some users to have unbounded privacy loss? We could achieve this by requiring the existence of a subset of users $\mathcal{T} \subseteq \mathcal{U}$ of density $1 - \delta$, such that every user in $\mathcal{T}$ is protected by ε-sketch privacy above cardinality N. In this case, a ratio δ of possible targets are not protected.

This approach only makes sense if the attacker cannot choose the target t. For our attacker model, this might be realistic: suppose that the attacker wants to target just one particular person. Since all user identifiers are hashed before being passed to the cardinality estimator, this person will be associated to a hash value that the attacker can neither predict nor influence. Thus, although the attacker picks t, the true target of the attack is $h(t)$, which the attacker cannot choose.

Unfortunately, this drastic increase in privacy risk for some users does not lead to a large increase in accuracy. Indeed, the best possible use of this ratio δ of users from an accuracy perspective would simply be to count exactly the users in a sample of sampling ratio δ.

Estimating the total cardinality based on this sample, similarly to the optimal estimator in the proof of Lemma 15, leads to a variance of $\frac{1-\delta}{\delta} \cdot (n - N)$. If δ is very small (say, $\delta \simeq 10^{-4}$), this variance is too large for counting small values of n (say, $n \simeq 1000$ and $N \simeq 100$). This is not surprising: if 99.99% of the values are ignored by the cardinality estimator, we cannot expect it to count values of n on the order of thousands. But even this value of δ is larger than what is often used with (ε, δ)-differential privacy, where typically, $\delta = o(1/n)$.

But in our running example, sketches must yield a reasonable accuracy both at small and large cardinalities, if many sketches are aggregated. This implicitly assumes that the service operates at a large scale, say with at least 10^7 users. With $\delta = 10^{-4}$, this means that thousands of users are not covered by the privacy property. This is unacceptable for most applications.

7.2 Averaging the privacy loss across users

Instead of requiring the same ε for every user, we could require that the *average* information gain by the attacker is bounded by ε. In this section, we take the example of HyperLogLog to show that accuracy is not incompatible with this notion of average privacy, but that cardinality estimators used in practice do not preserve privacy even if we average across all users.

First, we define this notion of average information gain across users.

Definition 29. Recall the definition of the positive privacy loss $\varepsilon_{n,t,M}$ of t given output M at cardinality n from Definition 24: The maximum privacy loss of t at cardinalty n is defined as $\varepsilon_{n,t} = \max_M (\varepsilon_{n,t,M})$. A cardinality estimator satisfies *ε-sketch privacy on average* if we have, for all n, $\frac{1}{|\mathcal{U}|} \sum_{t \in \mathcal{U}} \varepsilon_{n,t} \leq \varepsilon$.
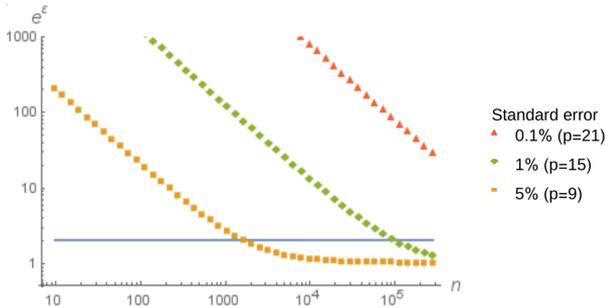
Fig. 3. ε_n as a function of n, for HyperLogLog cardinality estimators of different p parameters. The blue line is 2, corresponding to the commonly recommended value of $\varepsilon = \ln(2)$ in differential privacy.

In this definition, we accept that some users might have less privacy as long as the *average* user satisfies our initial privacy definition. Remark 28 is still relevant: we chose to average over all values of ε, but other averaging functions are possible and would lead to strictly stronger definitions.

We show that HyperLogLog satisfies this definition and we consider the value of ε for various parameters and their significance. Intuitively, a HyperLogLog cardinality estimator puts every element in a random *bucket*, and each bucket counts the *maximum number of leading zeroes* of elements added in this bucket. More details are given in Appendix E.

HyperLogLog cardinality estimators have a parameter p that determines its memory consumption, its accuracy, and, as we will see, its level of average privacy.

Theorem 30. *Assuming a sufficiently large $|\mathcal{U}|$, a HyperLogLog cardinality estimator of parameter p satisfies ε_n-sketch privacy above cardinality N on average where for $N \geq n$,*

$$\varepsilon_n \simeq -\sum_{k \geq 1} 2^{-k} \log\left(1 - \left(1 - 2^{-p-k}\right)^n\right).$$

The assumption that the set of possible elements is very large and its consequences are explained in more detail in the proof of this theorem, given in Appendix E.

How does this positive result fit practical use cases? Figure 3 plots ε_n for three different HyperLogLog cardinality estimators. It shows two important results.

First, cardinality estimators used in practice do not preserve privacy. For example, the default parameter used for production pipelines at Google and on the BigQuery service [2] is $p = 15$. For this value of p, an attacker can determine with significant accuracy whether a target was added to a sketch; not only in the worst case, but for the *average* user too. The average risk only becomes reasonable for $n \geq 10,000$, a threshold too large for most data analysis tasks.

Second, by sacrificing some accuracy, it is possible to obtain a reasonable *average privacy*. For example, a HyperLogLog sketch for which $p = 9$ has a relative standard error of about 5%, and an ε_n of about 1 for $n = 1000$. Unfortunately, even when the average risk is acceptable, some users will still be at a higher risk: users e with a large number of leading zeroes are much more identifiable than the average. For example, if $n = 1000$, there is a 98% chance that at least one user has $\rho(e) \geq 8$. For this user, $\varepsilon_{n,t} \simeq 5$, a very high value.

Our calculations yield only an approximation of ε_n that is an upper bound on the actual privacy loss in HyperLogLog sketches. However, these alarming results can be confirmed experimentally. We simulated $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right]$, for uniformly random values of t, using HyperLogLog sketches with the parameter $p = 15$, the default used for production pipelines at Google and on the BigQuery service [2]. For each cardinality n, we generated 10,000 different random target values, and added each one to 1,000 HyperLogLog sketches of cardinality n (generated from random values). For each target, we counted the number of sketches that ignored it.

Figure 4 plots some percentile values. For example, the all-targets curve (100th percentile) has a value of 33% at cardinality $n = 10,000$. This means that each of the 10,000 random targets was ignored by at most 33% of the 1,000 random sketches of this cardinality, i.e., $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \leq 33\%$ for all t. In other words, an attacker observes with at least 67% probability a change when adding a random target to a random sketch that did not contain it. Similarly, the 10th-percentile at $n = 10,000$ has a value of 3.8%. So 10% of the targets were ignored by at most 3.8% of the sketches, i.e., $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \leq 3.8\%$ for 10% of all users t. That is, for the average user t, there is a 10% chance that a sketch with 10,000 elements changes with likelihood at least 96.2% when t is first added.

For small cardinalities ($n < 1,000$), adding an element that has not yet been added to the sketch will almost certainly modify the sketch: an attacker observing that a sketch does not change after adding t can deduce with near-certainty that t was added previously.

Even for larger cardinalities, there is always a constant number of people with high privacy loss. For $n = 1,000$, no target was ignored by more than 5.5% of the sketches. For $n = 10,000$, 10% of the users were ignored by at most 3.8% of the sketches. Similarly, the 1st
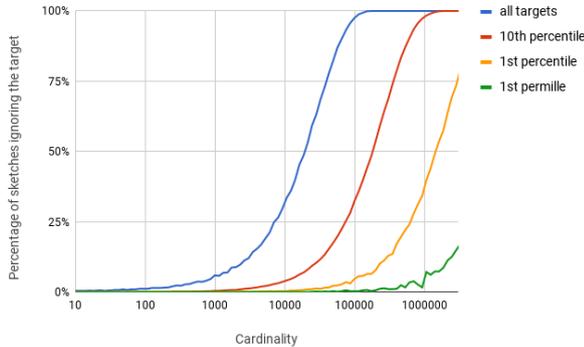
Fig. 4. Simulation of $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right]$, for uniformly chosen values of t, using HyperLogLog sketches with parameter $p = 15$.

percentile at $n = 100,000$ and the 1st permille at $n = 1,000,000$ are 4.6% and 4.5%, respectively. In summary, across all cardinalities n, at least 1,000 users t have $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \leq 0.05$. For these users, the corresponding privacy loss is $e^\varepsilon = \frac{1}{0.055} \simeq 18$. Concretely, if the attacker initially believes that $\mathbb{P}_n\left[t \in E\right]$ is 1%, this number grows to 15% after observing that $add\left(M_E, t\right) = M_E$. If it is initially 10%, it grows to 66%. And if it is initially 25%, it grows to 86%.

8 Mitigation strategies

A corollary of Theorem 12 and of our analysis of Section 7.2 is that the cardinality estimators used in practice do not preserve privacy. How can we best protect cardinality estimator sketches against insider threats, in realistic settings? Of course, classical data protection techniques are relevant: encryption, access controls, auditing of manual accesses, etc. But in addition to these best practices, cardinality estimators like HyperLogLog allow for specific risk mitigation techniques, which restrict the attacker's capabilities.

8.1 Salting the hash function with a secret

As explained in Section 3, most cardinality estimators use a hash function h as the first step of the *add* operation: $add\left(M, t\right)$ only depends on M and the hash value $h(t)$. This hash can be salted with a secret value. This salt can be made inaccessible to humans, with access controls restricting access to production binaries compiled from trusted code. Thus, an adversary cannot learn all the relevant parameters of the cardinality

estimator and can no longer add users to sketches. Of course, to avoid a salt reconstruction attack, a cryptographic hash function must be used.

The use of a salt does not hinder the usefulness of sketches: they can still be merged (for all cardinality estimators given as examples in Section 3) and the cardinality can still be estimated without accuracy loss. However, if an attacker gains direct access to a sketch M with the aim of targeting a user t and does not know the secret salt, then she cannot compute $h(t)$ and therefore cannot compute $add\left(M, t\right)$. This prevents the previous obvious attack of adding t to M and observing whether the result is different.

However, this solution has two issues.

Secret salt rotation The secret salt must be the same for all sketches as otherwise sketches cannot be merged. Indeed, if a hash function h_1 is used to create a sketch M_1 and h_2 is used to create M_2, then if $h_1(t) \neq h_2(t)$ for some t that is added to both M_1 and M_2, t will be seen as a *different user* in M_1 and M_2: the cardinality estimator no longer ignores duplicates. Good key management practices also recommend regularly rotating secret keys. In this context, changing the key requires recomputing all previously computed sketches. This requires keeping the original raw data, makes pipelines more complex, and can be computationally costly.

Sketch intersection For most cardinality estimators given as examples in Section 3, it is possible for an attacker to guess $h(t)$ from a family of sketches $(M_1, \ldots, M_k)$ for which the attacker *knows* that $t \in M_1$. For example, intersecting the lists stored in K-Minimum Values sketches can provide information on which hashes come from users that have been in *all* sketches. For HyperLogLog, one can use the leading zeroes in non-empty buckets to get partial information on the hash value of users who are in all sketches. Moreover, HyperLogLog++ [29] has a *sparse mode* that stores full hashes when the sketch contains a small number of values; this makes intersection attacks even easier.

Intersection attacks are realistic, although they are significantly more complex than simply checking if $add\left(M, t\right) = M$. In our running example, sketches come from counting users across locations and time periods. If an internal attacker wants to target someone she knows, she can gather information about where they went using side channels like social media posts. This gives her a series of sketches $M_1, \ldots, M_k$ that she *knows* her target belongs to, and from these, she can get information on $h(t)$ and use it to perform an attack equivalent to checking whether $add\left(M, t\right) = M$.

Another possible risk mitigation technique is homomorphic encryption. Each sketch could be encrypted in a way that allows sketches to be merged, and new elements to be added; while ensuring that an attacker cannot do any operation without some secret key. Homomorphic encryption typically has significant overhead, so it is likely to be too costly for most use cases. Our impossibility results assume a computationally unbounded attacker; however, it is possible that an accurate sketching mechanism using homomorphic encryption could provide privacy against *polynomial-time* attackers. We leave this area of research for future work.

8.2 Using a restricted API

Using cardinality estimator sketches to perform data analysis tasks only requires access to two operations: *merge* and *estimate*. So a simple option is to process the sketches over an API that only allows this type of operation. One option is to provide a SQL engine on a database, and only allow SQL functions that correspond to *merge* and *estimate* over the column containing sketches. In the BigQuery SQL engine, this corresponds to allowing HLL_COUNT.MERGE and HLL_-COUNT.EXTRACT functions, but not other functions over the column containing sketches [2]. Thus, the attacker cannot access the raw sketches.

Under this technique, an attacker who only has access to the API can no longer directly check whether $add(M, t) = M$. Since she does not have access to the sketch internals, she cannot perform the intersection attack described previously either. To perform the check, her easiest option is to impersonate her target within the service, interact with the service so that a sketch $M_{\{t\}}$ containing *only* her target is created in the sketch database, and compare the estimates obtained from M and $merge(M, M_{\{t\}})$. Following the intuition given in Section 5.1, if these estimates are the same, then the target is more likely to be in the dataset. How much information the attacker gets this way depends on $\mathbb{P}\left[estimate\left(add\left(M_E, t\right)\right) = estimate\left(M_E\right) \mid t \notin E\right]$. We can increase this quantity by rounding the result of the *estimate* operation, thus limiting the accuracy of the external attacker. This would make the attack described in this work slightly more difficult to execute, and less efficient. However, it is likely that the attack could be adapted, for example by repeating it multiple times with additional fake elements.

This risk mitigation technique can be combined with the previous one. The restricted API protects the sketches during normal use by data analysts, i.e., against external attackers. The hash salting mitigates the risk of manual access to the sketches, e.g., by internal attackers. This type of direct access is not needed for most data analysis tasks, so it can be monitored via other means.

9 Conclusion

We formally defined a class of cardinality estimator algorithms with an associated system and attacker model that captures the risks associated with processing personal data in cardinality estimator sketches. Based on this model, we proposed a privacy definition that expresses that the attacker cannot gain significant knowledge about a given target.

We showed that our privacy definition, which is strictly weaker than any reasonable definition used in practice, is incompatible with the accuracy and aggregation properties required for practical uses of cardinality estimators. We proved similar results for even weaker definitions, and we measured the privacy loss associated with the HyperLogLog cardinality estimator, commonly used in data analysis tasks.

Our results show that designing accurate privacy-preserving cardinality estimator algorithms is impossible, and that the cardinality estimator sketches used in practice should be considered as sensitive as raw data. These negative results are a consequence of the *structure* imposed on cardinality estimators: idempotence, commutativity, and existence of a well-behaved merge operation. This result shows a fundamental incompatibility between accurate aggregation and privacy. A natural question is ask whether other sketching algorithms have similar incompatibilities, and what are minimal axiomatizations that lead to similar impossibility results.

10 Acknowledgements

References

[1] [n. d.]. Approximate algorithms in Apache Spark: HyperLogLog and Quantiles. https://databricks.com/blog/2016/05/19/approximate-algorithms-in-apache-spark-hyperloglog-and-quantiles.html. Accessed: 2018-05-03.

[2] [n. d.]. BigQuery Technical Documentation on Functions & Operators. https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators. Accessed: 2017-10-24.

[3] [n. d.]. HyperLogLog implementation for mssql. https://github.com/shuvava/mssql-hll. Accessed: 2018-05-03.

[4] [n. d.]. PostgreSQL extension extension adding HyperLogLog data structures as a native data type. https://github.com/citusdata/postgresql-hll. Accessed: 2018-05-03.

[5] Vikas G Ashok and Ravi Mukkamala. 2014. A scalable and efficient privacy preserving global itemset support approximation using Bloom filters. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 382–389.

[6] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*. Springer, 1–10.

[7] Raef Bassily, Adam Groce, Jonathan Katz, and Adam Smith. 2013. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 439–448.

[8] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 127–135.

[9] Raef Bassily, Uri Stemmer, Abhradeep Guha Thakurta, et al. 2017. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*. 2288–2296.

[10] Kevin Beyer, Peter J Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. 2007. On synopses for distinct-value estimation under multiset operations. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 199–210.

[11] Raghav Bhaskar, Abhishek Bhowmick, Vipul Goyal, Srivatsan Laxman, and Abhradeep Thakurta. 2011. Noiseless database privacy. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 215–232.

[12] George Casella and Roger L Berger. 2002. *Statistical inference*. Vol. 2. Duxbury Pacific Grove, CA.

[13] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[14] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2002. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*. Springer, 54–68.

[15] Yitao Duan. 2009. Privacy without noise. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 1517–1520.

[16] Marianne Durand and Philippe Flajolet. 2003. Loglog counting of large cardinalities. In *European Symposium on Algorithms*. Springer, 605–617.

[17] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*. Springer, 1–19.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*. Springer, 265–284.

[19] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE, 51–60.

[20] Rolf Egert, Marc Fischlin, David Gens, Sven Jacob, Matthias Senker, and Jörn Tillmanns. 2015. Privately computing set-union and set-intersection cardinality via Bloom filters. In *Australasian Conference on Information Security and Privacy*. Springer, 413–430.

[21] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 1054–1067.

[22] Cristian Estan, George Varghese, and Mike Fisk. 2003. Bitmap algorithms for counting active flows on high speed links. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 153–166.

[23] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies* 2016, 3 (2016), 41–61.

[24] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2008. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings* 1 (2008).

[25] Philippe Flajolet and G Nigel Martin. 1985. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31, 2 (1985), 182–209.

[26] Philippe Golle and Kurt Partridge. 2009. On the anonymity of home/work location pairs. *Pervasive computing* (2009), 390–397.

[27] Michaela Gotz, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke. 2012. Publishing search logs—a comparative study of privacy guarantees. *IEEE Transactions on Knowledge and Data Engineering* 24, 3 (2012), 520–532.

[28] Krzysztof Grining and Marek Klonowski. 2017. Towards Extending Noiseless Privacy: Dependent Data and More Practical Approach. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 546–560.

[29] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*. ACM, 683–692.

[30] Daniel Kifer and Ashwin Machanavajjhala. 2012. A rigorous and customizable framework for privacy. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 77–88.

[31] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: Theory meets practice on the map. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. IEEE Computer Society, 277–286.

[32] Luca Melis, George Danezis, and Emiliano De Cristofaro. 2015. Efficient private statistics with succinct sketches. *arXiv preprint arXiv:1508.06110* (2015).

[33] Ilya Mironov. 2017. Renyi differential privacy. In *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*. IEEE, 263–275.

[34] Anna Monreale, Wendy Hui Wang, Francesca Pratesi, Salvatore Rinzivillo, Dino Pedreschi, Gennady Andrienko, and Natalia Andrienko. 2013. Privacy-preserving distributed movement data aggregation. In *Geographic Information Science at the Heart of Europe*. Springer, 225–245.

[35] Sriram Padmanabhan, Bishwaranjan Bhattacharjee, Tim Malkemus, Leslie Cranston, and Matthew Huras. 2003. Multidimensional clustering: A new data layout scheme in db2. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 637–641.

[36] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. 2010. Cardinality estimation and dynamic length adaptation for Bloom filters. *Distributed and Parallel Databases* 28, 2 (2010), 119–156.

[37] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. 2009. Relationship privacy: output perturbation for queries with joins. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 107–116.

[38] David Rebollo-Monedero and Jordi Forné. 2010. Optimized query forgery for private information retrieval. *IEEE Transactions on Information Theory* 56, 9 (2010), 4631–4642.

[39] David Rebollo-Monedero, Jordi Forne, and Josep Domingo-Ferrer. 2010. From t-closeness-like privacy to postrandomization via information theory. *IEEE Transactions on Knowledge and Data Engineering* 22, 11 (2010), 1623–1636.

[40] Amit Shukla, Prasad Deshpande, Jeffrey F Naughton, and Karthikeyan Ramasamy. 1996. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *VLDB*, Vol. 96. 522–531.

[41] Florian Tschorsch and Björn Scheuermann. 2013. An algorithm for privacy-preserving distributed user statistics. *Computer Networks* 57, 14 (2013), 2775–2787.

[42] Kyu-Young Whang, Brad T Vander-Zanden, and Howard M Taylor. 1990. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems (TODS)* 15, 2 (1990), 208–229.

[43] Shuheng Zhou, Katrina Ligett, and Larry Wasserman. 2009. Differential privacy with compression. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE, 2718–2722.

A Proof of Lemma 5

Let M and M' be two sketches. If $E = \{e_1, \ldots, e_k\}$, we denote by $add\,(M, E)$ the result of adding elements of E successively to M: $add\,(M, E) = add\,(\ldots add\,(add\,(M, e_1), e_2) \ldots, e_k)$.

Let E_1 and E_2 be two sets such that $M_{E_1} = M_{E_2} = M'$, and let E be a set such as $M_E = M$. Then $add\,(M, E_1) = add\,(add\,(M_\varnothing, E), E_1)$. Using the two properties of the add function, we get $add\,(M, E_1) = add\,(add\,(M_\varnothing, E_1), E) = add\,(M', E)$. The same reasoning leads to $add\,(M, E_2) = add\,(M, E_1)$: the merge function does not depend on the choice of E.

In addition, note that $merge\,(M, M') = add\,(M_\varnothing, E \cup E_1)$. Thus, commutativity, associativity, idempotence, and neutrality follow directly from the same properties of set union.

B Full proof of Lemma 15

The proof is decomposed into three steps. In Step 1, we fix the first N elements added to the sketch, and we bound the variance of the estimator that has these elements as initial input. In Step 2, we explicitly compute the probability for an element t to be ignored by M_E, first when t is fixed, then when E is fixed. We then use these results in Step 3 to average the bound over all possible choices for the N elements in E, which gives us an overall bound.

Step 1: Bound the estimator variance.

Let $E \in \mathcal{P}_N\,(\mathcal{U})$. Let $S_E = \{t \in \mathcal{U} \mid add\,(M_E, t) \neq M_E\}$ be the set of elements that are *not* ignored by M_E. Let $p_E = \frac{|S_E|}{|\mathcal{U}|}$, or equivalently, let $p_E = \mathbb{P}_t\,[add\,(M_E, t) \neq M_E \mid t \notin E]$, where $\mathbb{P}_t$ is the distribution that picks t uniformly randomly in $\mathcal{U}$.

S_E can be seen as the *sampling set* of the estimator with E as initial input, while p_E can be seen as its *sampling fraction*: all other elements are discarded, so the estimator only has access to elements in E and S_E to compute its estimate.

The optimal estimator to minimize variance in this context simply counts the exact number of distinct elements in the sample (remembering each one), and di-

vides this number by p_E to estimate the total number of distinct elements.[1]

What is the variance $\mathbb{V}_{n|E}$ of this optimal estimator? Suppose we added $n - N$ elements after reaching the sampling part. The number of elements in the sample is a random variable with variance $p_E (1 - p_E) (n - N)$. Dividing this random variable by p_E gives a variance of $\frac{1 - p_E}{p_E} \cdot (n - N)$. Thus

$$\mathbb{V}_{n|E} \geq \frac{1 - p_E}{p_E} \cdot (n - N).$$

Since the first N elements are chosen uniformly at random, the variance of the overall estimator is bounded by their average. If we denote by $\mathcal{P}_N(\mathcal{U})$ the set of all possible subsets $E \subseteq \mathcal{U}$ of cardinality N, we have:

$$\mathbb{V}_n \geq \mathrm{avg}_{E \in \mathcal{P}_N(\mathcal{U})} \frac{1 - p_E}{p_E} (n - N) \qquad (1)$$

(where avg stands for the average).

Step 2: Intermediary results.

Fix $E \in \mathcal{P}_N(\mathcal{U})$. Denoting by 1_X the function whose value is 1 if X is satisfied and 0 otherwise,

$$\mathbb{P}_t \left[add \left(M_E, t \right) \neq M_E \mid t \notin E \right]$$
$$= \frac{\mathbb{P}_t \left[add \left(M_E, t \right) \neq M_E \wedge t \notin E \right]}{\mathbb{P}_t \left[t \notin E \right]}$$
$$= \frac{|\mathcal{U}|}{|\mathcal{U}| - N} \cdot \mathrm{avg}_{t \in \mathcal{U}} \left(1_{add(M_E, t) \neq M_E} \right). \qquad (2)$$

Indeed, $\mathbb{P}_t \left[t \notin E \right] = \frac{|\mathcal{U}| - N}{|\mathcal{U}|}$ is straightforward, and since $t \in E$ implies $add \left(M_E, t \right) = M_E$, the condition $add \left(M_E, t \right) \neq M_E \wedge t \notin E$ can be simplified to $add \left(M_E, t \right) \neq M_E$.

Now, fix $t \in \mathcal{U}$. Then

$$\mathbb{P}_N \left[add \left(M_E, t \right) \neq M_E \mid t \notin E \right]$$
$$= \frac{\mathbb{P}_N \left[add \left(M_E, t \right) \neq M_E \wedge t \notin E \right]}{\mathbb{P}_N \left[t \notin E \right]}$$
$$= \frac{|\mathcal{U}|}{|\mathcal{U}| - N} \cdot \mathrm{avg}_{E \in \mathcal{P}_N(E)} \left(1_{add(M_E, t) \neq M_E} \right). \qquad (3)$$

Indeed, we have

$$\mathbb{P}_N \left[t \notin E \right] = \binom{|\mathcal{U}| - 1}{N} \cdot \binom{|\mathcal{U}|}{N}^{-1}$$
$$= \frac{(|\mathcal{U}| - 1)!}{(|\mathcal{U}| - 1 - N)! N!} \cdot \frac{(|\mathcal{U}| - N)! N!}{|\mathcal{U}|!} = \frac{|\mathcal{U}| - N}{|\mathcal{U}|}$$

Step 3: Conclude using convexity.

We now prove that $\mathrm{avg}_{E \in \mathcal{P}_N(\mathcal{U})} (p_E) \leq p$. Our initial hypothesis states that $p \geq \mathbb{P}_N \left[add \left(M_E, t \right) \neq M_E \mid t \notin E \right]$ for all t. We can average this for every t and use (3) and (2):

$$p \geq \mathrm{avg}_{t \in \mathcal{U}} \left(\mathbb{P}_N \left[add \left(M_E, t \right) \neq M_E \mid t \notin E \right] \right)$$
$$\overset{(3)}{=} \mathrm{avg}_{t \in \mathcal{U}} \left(\frac{|\mathcal{U}|}{|\mathcal{U}| - N} \cdot \mathrm{avg}_{E \in \mathcal{P}_N(E)} \left(1_{add(M_E, t) \neq M_E} \right) \right)$$
$$= \mathrm{avg}_{E \in \mathcal{P}_N(\mathcal{U})} \left(\frac{|\mathcal{U}|}{|\mathcal{U}| - N} \cdot \mathrm{avg}_{t \in \mathcal{U}} \left(1_{add(M_E, t) \neq M_E} \right) \right)$$
$$\overset{(2)}{=} \mathrm{avg}_{E \in \mathcal{P}_N(\mathcal{U})} \left(\mathbb{P}_t \left[add \left(M_E, t \right) \neq M_E \mid t \notin E \right] \right)$$
$$\geq \mathrm{avg}_{E \in \mathcal{P}_N(\mathcal{U})} (p_E) \qquad (4)$$

Now, using (1) and (4) along with the fact that the function $x \longrightarrow \frac{1 - x}{x} \cdot (n - N)$ is convex and decreasing on $(0, 1)$, we can conclude by Jensen's inequality that

$$\mathbb{V}_n \geq \frac{1 - p}{p} (n - N).$$

C Proof of Lemma 22

First, we show that if a cardinality estimator verifies (ε, δ)-sketch privacy at a given cardinality, then for each target t, we can find an explicit decomposition of the possible outputs: the bound on information gain is satisfied for each possible output, except on a density δ. This is similar to the definition of *probabilistic* differential privacy in [31] and [27], except the decomposition depends on the choice of t. We then use this decomposition to prove a variant of our negative result.

Lemma 31. *If a cardinality estimator satisfies (ε, δ)-sketch privacy above cardinality N, then for every $n \geq N$ and $t \in \mathcal{U}$, we can decompose the space of possible sketches $\mathcal{M} = \mathcal{M}_1 \uplus \mathcal{M}_2$ such that $\mathbb{P}_n \left[M_E \in \mathcal{M}_2 \mid t \in E \right] \leq 2\delta$, and for all $M \in \mathcal{M}_1$:*

$$\mathbb{P}_n \left[M_E = M \mid t \in E \right] \leq 2e^\varepsilon \cdot \mathbb{P}_n \left[M_E = M \mid t \notin E \right].$$

Proof. Suppose the cardinality estimator satisfies (ε, δ)-sketch privacy at cardinality N, and fix $n \geq N$ and $t \in \mathcal{U}$. Let $\varepsilon' = \varepsilon + \ln(2)$.

Let $\mathcal{M}_1$ be the set of outputs for which the privacy loss is higher than ε'. Formally, $\mathcal{M}_1$ is the set of sketches M that satisfy

$$\mathbb{P}_n \left[M_E = M \mid t \in E \right] > e^{\varepsilon'} \cdot \mathbb{P}_n \left[M_E = M \mid t \notin E \right].$$

We show that $\mathcal{M}_1$ has a density bounded by 2δ.

Suppose for the sake of contradiction that this set has density at least 2δ given that $t \in E$:

$$\mathbb{P}_n\left[M \in \mathcal{M}_1 \mid t \in E\right] \geq 2\delta.$$

We can sum the inequalities in $\mathcal{M}_1$ to obtain

$$\mathbb{P}_n\left[M_E \in \mathcal{M}_1 \mid t \in E\right] > e^{\varepsilon'} \cdot \mathbb{P}_n\left[M_E \in \mathcal{M}_1 \mid t \notin E\right].$$

Averaging both inequalities, we get

$$\mathbb{P}_n\left[M_E \in \mathcal{M}_1 \mid t \in E\right] > e^{\varepsilon} \cdot \mathbb{P}_n\left[M_E \in \mathcal{M}_1 \mid t \notin E\right] + \delta$$

since $e^{\varepsilon'} = e^{\varepsilon + \ln(2)} = 2 \cdot e^{\varepsilon}$. This contradicts the hypothesis that the cardinality estimator satisfied (ε, δ)-sketch privacy at cardinality N.

Thus, $\mathbb{P}_n\left[M \in \mathcal{M}_1 \mid t \in E\right] < 2\delta$ and by the definition of $\mathcal{M}_1$, every output $M \in \mathcal{M}_2 = \mathcal{M} \setminus \mathcal{M}_1$ verifies:

$$\mathbb{P}_n\left[M_E = M \mid t \in E\right] \leq e^{\varepsilon'} \cdot \mathbb{P}_n\left[M_E = M \mid t \notin E\right].$$

$\square$

We can then use this decomposition to prove Lemma 22. Lemma 31 allows us to get two sets $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

– $\mathbb{P}_n\left[M_E \in \mathcal{M}_2 \mid t \in E\right] \leq 2\delta$; and
– for all $M \in \mathcal{M}_1$:

$$\mathbb{P}_n\left[M_E = M \mid t \in E\right] \leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[M_E = M \mid t \notin E\right].$$

We decompose $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \in E\right]$ into

$$\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \wedge M_E \in \mathcal{M}_1 \mid t \in E\right]$$
$$+ \mathbb{P}_n\left[add\left(M_E, t\right) = M_E \wedge M_E \in \mathcal{M}_2 \mid t \in E\right].$$

The same reasoning as in the proof of Lemma 13 gives

$$\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \wedge M_E \in \mathcal{M}_1 \mid t \in E\right]$$
$$\leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[add\left(M_E, t\right) = M_E \wedge M_E \in \mathcal{M}_1 \mid t \notin E\right]$$
$$\leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right]$$

and since $\mathbb{P}_n\left[M_E \in \mathcal{M}_2 \mid t \in E\right] \leq 2\delta$, we immediately have

$$\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \wedge M \in \mathcal{M}_2 \mid t \in E\right] \leq 2\delta.$$

We conclude that

$$\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \in E\right]$$
$$\leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] + 2\delta.$$

Now, Lemma 3 gives $\mathbb{P}\left[add\left(M_E, t\right) = M_E \mid t \in E\right] = 1$, and finally $\mathbb{P}_n\left[add\left(M_E, t\right) = M_E \mid t \notin E\right] \geq \left(\frac{1}{2} - \delta\right) \cdot e^{-\varepsilon}$.

D Proof of Lemma 26

Let $n \geq N$ and $\delta > 0$. Suppose that a cardinality estimator does not satisfy $\left(\frac{\varepsilon}{\delta}, \delta\right)$-sketch privacy at cardinality n. Then with probability strictly larger than δ, the output does not satisfy $\frac{\varepsilon}{\delta}$-sketch privacy. Formally there is $\mathcal{M}_2 \subseteq \mathcal{M}$ such that $\mathbb{P}_n\left[M_E \in \mathcal{M}_2 \mid t \in E\right] > \delta$, and such that $\varepsilon_{n,t,M} > \frac{\varepsilon}{\delta}$ for all $M \in \mathcal{M}_2$. Since all values of $\varepsilon_{n,t,M}$ are positive, we have

$$\sum_M \mathbb{P}_n\left[M_E = M \mid t \in E\right] \cdot \varepsilon_{n,t,M}$$
$$\geq \sum_{M \in \mathcal{M}_2} \mathbb{P}_n\left[M_E = M \mid t \in E\right] \cdot \varepsilon_{n,t,M}$$
$$> \frac{\varepsilon}{\delta} \sum_{M \in \mathcal{M}_2} \mathbb{P}_n\left[M_E = M \mid t \in E\right]$$
$$> \varepsilon.$$

Hence this cardinality estimator does not satisfy ε-sketch average privacy at cardinality n.

E Proof of Theorem 30

First, let us formally define a HyperLogLog cardinality estimator.

Definition 32. Let h be a uniformly distributed hash function. A *HyperLogLog cardinality estimator of parameter p* is defined as follows. A sketch consists of a list of 2^p counters $C_0, \ldots, C_{2^p - 1}$, all initialized to 0. When adding an element e to the sketch, we compute $h(e)$, and represent it as a binary string $x = x_1 x_2 \ldots$. Let $b(e) = \langle x_1 \ldots x_p \rangle_2$, i.e., the integer represented by the binary digits $x_1 \ldots x_p$, and $\rho(e)$ be the position of the leftmost 1-bit in $x_{p+1} x_{p+2} \ldots$. Then we update counter $C_{b(e)}$ with $C_{b(e)} \leftarrow \max\left(C_{b(e)}, \rho(e)\right)$.

For example, suppose that $x = 10001101\ldots$ and $p = 2$, then $b(e) = \langle 10 \rangle_2 = 2$, and $\rho(e) = 3$ (the position of the leftmost 1-bit in $001101\ldots$). So we must look at the value for the counter C_2 and, if $C_2 < 3$, set C_2 to 3.

To simplify our analysis, we assume in this proof that $|\mathcal{U}|$ is very large: for all reasonable values of n, picking n elements uniformly at random in $\mathcal{U}$ is the same as picking a subset of $\mathcal{U}$ of size n uniformly at random. In particular, we approximate $\mathbb{P}_n\left[M_E = M \mid t \notin E\right]$ by $\mathbb{P}_n\left[M_E = M\right]$.

First, we compute $\varepsilon_{n,t,M}$ for each M, by considering the counter values of M. We then use this to determine $\varepsilon_{n,t}$, and averaging them, deduce the desired result.

Step 1: Computing $\varepsilon_{n,t,M}$

Let $t \in \mathcal{U}$, and $M \in \mathcal{M}$ such that $\mathbb{P}_n\left[M_E = M \mid t \in E\right] > 0$. Decompose the binary string $h\left(t\right) = x$ into two parts to get $b\left(t\right)$ and $\rho\left(t\right)$. Denote by $C_0, \ldots, C_{2^p-1}$ the counters of M. Note that $M = M_E$ is characterized by two conditions:

- $\text{REACH}_E(i)$ for all $0 \leq i < 2^p$, where $\text{REACH}_E(i)$ abbreviates $C_i > 0 \longrightarrow \exists e \in E.\ b(e) = i \wedge \rho(e) = C_i$. For each non-empty bucket, an element with this number of leading zeroes was added in this bucket.
- $\forall e \in E.\ \rho(e) \leq C_{b(e)}$ (notation MAX_E). No element has more leading zeroes than the counter for its bucket.

Now, we compute the value of $\varepsilon_{n,t,M}$, depending on the value of $C_{b(t)}$. Without loss of generality, we assume that $b(t) = 0$.

Case 1: Suppose $C_0 = \rho(t)$.

We compute the probability of observing M given $t \in E$. $\text{REACH}_E(0)$ is already satisfied by t as $C_0 = \rho(t)$. So for M_E to be equal to M, E's other elements only have to satisfy $\text{REACH}(i)$ for $i \geq 1$, and MAX_E for all i:

$$\mathbb{P}_n[M_E = M \mid t \in E] = \mathbb{P}_{n-1}[\text{MAX}_E \wedge \forall i \geq 1.\ \text{REACH}_E(i)].$$

Next, we compute the probability of observing M given $t \notin E$. This time, all elements of E are chosen randomly, so

$$\mathbb{P}_n\left[M_E = M \mid t \notin E\right] \simeq \mathbb{P}_n\left[M_E = M\right]$$
$$= \mathbb{P}_n\left[\text{MAX}_E \wedge \forall i.\ \text{REACH}_E(i)\right].$$

We can decompose this condition: there is a witness $e \in E$ for $\text{REACH}_E(0)$ (i.e., $b(e) = 0$ and $\rho(e) = C_0 = \rho(t)$) and all others elements satisfy the same condition as in the case $t \in E$, namely $\text{MAX}_E \wedge \forall i \geq 1.\ \text{REACH}_E(i)$, as this is equivalent to $\text{MAX}_{E \setminus \{e\}} \wedge \forall i \geq 1.\ \text{REACH}_{E \setminus \{e\}}(i)$ by the choice of e. If e is chosen uniformly in $\mathcal{U}$, then

- The probability of $b(e) = 0$ is 2^{-p}, since h is a uniformly distributed hash function.
- The probability of $\rho(e) = \rho(t)$ is $2^{-\rho(t)}$: since h is a distributed hash function, if we note $h(t) = x_1 \ldots x_p.x_{p+1} \ldots$, then $x_{p+1} = 1$ with probability $1/2$, $x_{p+1}x_{p+2} = 01$ with probability $1/4$, etc.

Thus, the probability that an element e witnesses $\text{REACH}_E(0)$ is $2^{-p-\rho(t)}$ as $x_1 \ldots x_p$ are independent of $x_{p+1}x_{p+2} \ldots$. Since the set E has size n, there are n possible chances that such an element is chosen: the probability that at least one element witnesses

$\text{REACH}_E(0)$ is $1 - \left(1 - 2^{-p-\rho(t)}\right)^n$. We can thus approximate $\mathbb{P}_n\left[M_E = M \mid t \notin E\right]$ by

$$\left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right) \cdot \mathbb{P}_{n-1}\left[\text{MAX}_E \wedge \forall i \geq 1.\ \text{REACH}_E(i)\right]$$

and thus

$$\frac{\mathbb{P}_n\left[M_E = M \mid t \in E\right]}{\mathbb{P}_n\left[M_E = M \mid t \notin E\right]} \simeq \left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right)^{-1}.$$

Case 2: Suppose $C_0 > \rho(t)$.

We can compute the probabilities $\mathbb{P}_n\left[M_E = M \mid t \in E\right]$ and $\mathbb{P}_n\left[M_E = M \mid t \notin E\right]$ in a similar fashion.

$$\mathbb{P}_n\left[M_E = M \mid t \in E\right] = \mathbb{P}_{n-1}\left[\text{MAX}_E \wedge \forall i.\ \text{REACH}_E(i)\right]$$

$$\mathbb{P}_n\left[M_E = M \mid t \notin E\right] \simeq \mathbb{P}_n\left[M_E = M\right]$$
$$= \mathbb{P}_n\left[\text{MAX}_E \wedge \forall i.\ \text{REACH}_E(i)\right].$$

Since (by hypothesis) $\mathbb{P}_n\left[M_E = M \mid t \in E\right] > 0$, there exist $n - 1$ distinct elements of $\mathcal{U}$ which, together, satisfy this condition $\text{MAX}_E \wedge \forall i.\ \text{REACH}_E(i)$. This allows us to bound $\mathbb{P}_n\left[M_E = M\right]$ from below. Suppose one element e of E satisfies $b(e) = 0$ and $\rho(e) = \rho(t)$, and the $n - 1$ other elements in $E \setminus \{e\}$ satisfy the conditions $\text{MAX}_{E \setminus \{e\}}$ and $\text{REACH}_{E \setminus \{e\}}(i)$ for all i. Then E satisfies MAX_E and $\text{REACH}_E(i)$ for all i. The lower bound follows as before:

$$\mathbb{P}_n\left[M_E = M \mid t \notin E\right] \underset{\sim}{\geq}$$
$$\left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right) \cdot \mathbb{P}_{n-1}\left[\text{MAX}_E \wedge \forall i.\ \text{REACH}_E(i)\right]$$

and thus

$$\frac{\mathbb{P}_n\left[M_E = M \mid t \in E\right]}{\mathbb{P}_n\left[M_E = M \mid t \notin E\right]} \underset{\sim}{\lesssim} \left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right)^{-1}.$$

We can use the results of both cases and immediately conclude that

$$\varepsilon_{n,t,M} \underset{\sim}{\lesssim} -\log\left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right)$$

and that the equality holds if M satisfies $C_{b(t)} = \rho(t)$.

Step 2: Determining ε_n

The previous reasoning shows that the worst case happens when the counter corresponding to t's bucket contains the number of leading zeroes of t, i.e., when $\rho(t) = C_{b(t)}$:

$$\varepsilon_{n,t} = \max_M\left(\varepsilon_{n,t,M}\right) \simeq -\log\left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right).$$

We can then average this value over all t. Since the hash function is uniformly distributed, $1/2$ of the values of t satisfy $\rho(t) = 1$, $1/4$ satisfy $\rho(t) = 1/4$, etc. Thus

$$\varepsilon_n \simeq -\sum_{k \geq 1} 2^{-k} \log\left(1 - \left(1 - 2^{-p-k}\right)^n\right).$$