

Privacy via Pseudorandom Sketches

Nina Mishra*

Mark Sandler†

December 18, 2005

Abstract

Imagine a collection of individuals who each possess private data that they do not wish to share with a third party. This paper considers how individuals may represent and publish their own data so as to simultaneously preserve their privacy and to ensure that it is possible to extract large-scale statistical behavior from the original unperturbed data. Existing techniques for perturbing data are limited by the number of users required to obtain approximate answers to queries, the richness of preserved statistical behavior, the privacy guarantees given and/or the amount of data that each individual must publish.

This paper introduces a new technique to describe parts of an individual's data that is based on pseudorandom sketches. The sketches guarantee that each individual's privacy is provably maintained assuming one of the strongest definitions of privacy that we are aware of: given unlimited computational power and arbitrary partial knowledge, the attacker can not learn any additional private information from the published sketches. However, sketches from multiple users that describe a subset of attributes can be used to estimate the fraction of users that satisfy any conjunction over the full set of negated or unnegated attributes. We show that the error of approximation is independent of the number of attributes involved and only depends on the number of users available. An additional benefit is that the size of the sketch is minuscule: $\lceil \log \log O(M) \rceil$ bits, where M is the number of users. Finally, we show how sketches can be combined to answer more complex queries.

*University of Virginia, nmishra@cs.virginia.edu. Part of this work was done at HP Labs and Stanford University and was supported in part by NSF grant EIA-0137661.

†Cornell University, sandler@cs.cornell.edu. Part of this work was done at HP labs.

1 Introduction

Privacy is a paramount concern in applications where sensitive personal information is used for the purpose of discovering patterns. This data is often gathered by a central third party either with the assurance that each individual’s privacy will be maintained or with the goal of selling this information to interested buyers. In the former case, private data is not necessarily kept private as there have been many instances where organizations have either accidentally or intentionally violated their privacy agreements in the face of mergers, bankruptcy or theft [14, 15, 16]. In the latter case, private data is not even intended to be kept private. For example, Acxiom [17] is a company that sells private data to companies who hope to improve target marketing, customer retention, etc.

Since it is evident that our data is not private when trusted to another party, this paper develops an idea suggested in [1] – how can privacy be put back in the hands of individuals? Our view is that individuals maintain *all* of their private data and what we investigate are methods by which individuals can release perturbed versions of their personal data so that privacy is preserved and so that large-scale statistical patterns present in the original unperturbed data can be approximately recovered from the released perturbed set.

Some solutions to the non-trusted party problem have been proposed. One solution, known as *randomized response* advocated by Warner in the 1960s [24], amounts essentially to flipping bits in the private data. In this context, when an individual is surveyed about a sensitive matter, such as whether they ever inhaled, the respondent flips the answer to their question with probability p or answers truthfully with probability $1 - p$. If p is slightly less than $1/2$ then this simple bit-flipping procedure can be shown to preserve privacy and, with some simple analysis, if there are enough users then one can derive a reasonably good estimate of the actual fraction of people who answered the question yes. This bit flipping procedure, while powerful, has the disadvantage that if a user has a relatively sparse private vector then the resulting perturbed vector may be quite dense, and furthermore, the number of users required to compute more complicated queries grows very fast.

A more recent solution suggested by Evfimievski et al [10, 11] develops an improved randomized procedure that has similar privacy guarantees to bit flipping but also produces a compressed version of the dense flipped vector. However, their result only applies to databases where each user has a small number of items in their transaction (e.g. only small number of bits is set to one). Further, no analysis is given of the number of users needed to obtain accurate estimates of the frequency of an itemset as the size of the itemset grows. It appears (to us) that the number of users needed grows exponentially with the size of the itemset. This makes the method of only limited utility when users have a large number of bits set to 1, e.g., various poll data or non-binary data.

A generalization of randomized response to non-binary attributes was undertaken by Agrawal et al [3]. In this work, bit flipping was generalized to non-binary attributes via *retention replacement* – each user keeps their true value with fixed probability, or replaces their true value with noise. Arbitrary queries involving a fixed number of attributes can be answered with this technique. However, it has the disadvantage that an attacker with prior knowledge could learn a lot of information about a user. For example, if an attacker knows that someone’s private value is either $\langle 1, 1, 2, 2, 3, 3 \rangle$ or $\langle 4, 4, 5, 5, 6, 6 \rangle$ then seeing the perturbed sequence $\langle 1, 9, 8, 2, 3, 5 \rangle$ virtually reveals to the attacker the exact private data of the individual with high probability.

In this paper, we introduce a novel method where each user publishes a succinct representation of their data that we call a *sketch*. The sketch is derived using the theory of pseudorandom functions. Each sketch describes a subset of attributes over the user’s profile. We show that sketches preserve privacy in a very strong sense. However if such a sketch is collected from enough users then good estimates of the fraction of users that satisfy any conjunction over the full set of both negated and unnegated attributes can be obtained. The key difference from [10] is that the approximation error is *independent* of the number of attributes involved (as opposed to what seems to be exponential in the number of attributes in [10]).

Our Contributions To understand our contributions, we begin by defining privacy. If a user holds some private data and wishes to release a sketch s then we say that a user’s privacy is preserved if for *any* two possible private values \mathbf{d}' and \mathbf{d}'' , the attacker cannot distinguish between whether the original private data

was \mathbf{d}' or \mathbf{d}'' , i.e., $Pr(\mathbf{s}|\mathbf{d}') \approx Pr(\mathbf{s}|\mathbf{d}'')$. A more formal definition and discussion can be found in Section 2.¹

With regard to utility, the basic query that we strive to accurately answer is a *conjunctive query*: Suppose that each user holds a bit vector in $\mathbf{d} \in \{0, 1\}^q$ over the variables x_1, \dots, x_q . Given a subset of literals, where each element is either an attribute x_i or its negation \bar{x}_i , the output is the fraction of users that satisfy the conjunction. One example of a query is what fraction of individuals are HIV+ and do not have AIDS.

Our main contribution is a novel method where users may “publish” their data using succinct representations - *sketches*. Each sketch describes a subset of a user’s attributes which when collected from multiple users can be combined in order to approximate the fraction of users who satisfy an *arbitrary* conjunction. The key difference from [10] and [24] is that the error in estimating the answer to a conjunctive query is *independent* of the number of attributes involved. (This is in contrast to previous work, where the error appears to grow exponentially in the number of attributes in the query.) The size of the sketch is tiny: $\lceil \log \log O(M) \rceil$ where M is the number of users. We prove that sketches preserve privacy in the sense just described.

While we limit our analysis to only conjunctive queries, we demonstrate that these queries are quite powerful in the sense that many other queries (such as means, higher moments and interval queries) can be expressed as a collection of a small number of conjunctive queries.

Related Work In recent years several different approaches to privacy have emerged. They can be roughly divided into three categories by the extent to which the trusted third party is used in maintaining privacy. More specifically, they can be partitioned based on whether the trusted third party is (a) used to answer queries about the private data or (b) used only to create an initial sanitization of the private data or (c) not needed at all.

In the first approach, the idea is to perturb the answer to every query [5, 7, 9], or to precisely answer some queries but deny the answer to others [18, 19, 8] so as to ensure privacy. The second kind of work (that assumes a centralized third party who publishes a sanitized version of the data) includes different flavors of anonymity [20, 2, 23, 22] and data perturbation [6]. Note that in this case, the trusted server is needed only during the initial stage. After the data is released it immediately becomes public and no third party is needed to answer queries.

The most general approach, which is most relevant to our work does not assume the existence of a trusted party at all. Information provided by the user immediately becomes public and available for everybody’s use. The main idea here is to randomize or clean the user’s data [10, 11, 3] so that personal data is virtually hidden and yet it is possible to gather relevant statistics from the data of many individuals. To the best of our knowledge, no previous results could both operate on non-binary data and guarantee strong privacy.

Note that any result in the last two categories immediately implies a similar result in the first one. In particular, we believe that this paper provides useful insights for the framework described in [5]. We elaborate more on this in the Appendix A.

2 Preliminaries

Let \mathbf{d} denote a user’s private data which we sometimes refer to as a user’s *profile*. In addition we assume that each user holds a unique public identifier id - which does not contain any private information (for example it could be a timestamp of user registration in the system). The pair (id, \mathbf{d}) fully describe an individual. We denote the set of all pairs (id, \mathbf{d}) by \mathcal{D} .

Our goal is to provide the user with some mechanism for generating a sketch s about their profile so that user’s privacy is preserved. Informally, we say that the sanitized information preserves privacy if for the private data \mathbf{d}' and \mathbf{d}'' of any two individuals, $\Pr[\mathbf{s}|\mathbf{d}'] \approx \Pr[\mathbf{s}|\mathbf{d}'']$. Thus the sanitized data s does not help the attacker distinguish between the case when the user’s private data is \mathbf{d}' and \mathbf{d}'' . This definition is identical to [10]’s notion of γ -amplification.

Definition 1 *The sanitization \mathbf{s} is ϵ -private if for any two values of private data \mathbf{d}' and \mathbf{d}''*

$$\frac{\Pr[\text{User's sanitized value is } \mathbf{s} | \text{User's private value is } \mathbf{d}']}{\Pr[\text{User's sanitized value is } \mathbf{s} | \text{User's private value is } \mathbf{d}'']} \leq (1 + \epsilon) \tag{1}$$

¹In [10], this definition is known as γ -amplification.

Note that this definition is extremely strong since it says that the sanitized data is almost equally likely to be generated from *any* possible value of the underlying user data. Thus, no matter how much the attacker knows about a user in advance, after the sanitized sketch is released, very little new information can be learned. There are also other definitions of privacy, we present a brief comparison of them in Appendix C.

To better understand the definition of ϵ -privacy we give a simple example. Consider the situation where each user holds a single bit that is either 0 or 1. Then for $v \in \{0, 1\}$, we want

$$\frac{\Pr(\tilde{x}_i = v | x_i = 0)}{\Pr(\tilde{x}_i = v | x_i = 1)} \leq (1 + \epsilon) \quad (2)$$

Observe that if each individual flips their bit with probability exactly 1/2 then we have absolute privacy, but absolutely no utility. To understand why, observe that each user publishes 1 with probability 1/2 and 0 with probability 1/2 *independent* of their original unperturbed value. We do not even need a user to exhibit this kind of random behavior – all we need is a fair coin. Consequently, flipping with probability 1/2 does not provide any utility. However, if each individual flips their bit with probability p just a tinge under 1/2, i.e., $p = 1/2 - \epsilon$ then we can simultaneously ensure privacy and estimate the fraction of ‘1’s in the unperturbed data. The privacy proof is folklore so we have included it in Appendix B. Furthermore, the fraction of ones r can be estimated by solving for r in the following equation where \tilde{r} is the fraction of ones in the perturbed data: $E(\tilde{r}) = (1 - p)r + p(1 - r)$. (And, $E(\tilde{r})$ can be estimated using the Chernoff bound – refer to the proof of Lemma 4.1 for details.)

Conjunctive queries Conjunctive queries are a natural generalization of frequent item-set mining [10]. Given a subset of bits $B = \{b_1, \dots, b_k\}$, and their values $(\mathbf{v}_1 \dots \mathbf{v}_k)$ a conjunctive query returns the fraction of users that satisfy a query of the form $\bigwedge (\mathbf{d}_{b_i} = \mathbf{v}_i)$. In other words, given a set B and a binary string $\mathbf{v} = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, we want to estimate how many users satisfy $\mathbf{d}_B = \mathbf{v}$, where \mathbf{d}_B denotes a substring induced by set B . This generalization (from the monotone version considered in frequent itemset mining) enables efficient computation of a broad range of other queries - not necessarily on binary data. We let $I(B, \mathbf{v})$ denote the total number of users whose profile \mathbf{d} satisfies the constraint $\mathbf{d}_B = \mathbf{v}$. One way to compute answers to conjunctive queries, is via a system of linear equations similar to the one introduced in [10]. However the error introduced seems to grow exponentially in the number of bits involved and thus only appears to be useful for answering short, monotone conjunctive queries.

3 Sketches

In this section we develop an approach that will allow us to answer conjunctive queries defined on a large (or really, any size) subset of bits. The idea is that any subset of interest B can be *sketched*, so that we can answer the query $|I(B, \mathbf{v})|$ for an arbitrary value \mathbf{v} . In other words, each sketch over a set of k attributes gives us the ability to answer 2^k conjunctive queries (over the full set of k attributes, where each attribute appears either negated or unnegated). The sketching technique turns out to be very useful in mining non-binary data where for each attribute there are only a few subsets that need to be sketched. Sketches can also be combined together to produce answers to more complex queries.

Sketching can be viewed as an analog of hashing but with better privacy protection. Indeed, if each user hashes their value on a subset of bits B , then the hash value can be used to answer the query $I(B, \mathbf{v})$, by computing the number of users who hold the hash value of vector \mathbf{v} . However, even though the hash function is non-reversible, it might violate privacy. Indeed, if Bob knows that Alice’s private value can be only one out of 100 known possible values, then once he sees the hash value, by applying the hash function to each potential value, he can deduce the original value (with very high probability). Sketches are devoid of this property – seeing a sketch tells almost nothing (in a precise mathematical sense) about the private value which generated it.

Intuition Consider a subset of bits B . We want to be able to estimate the fraction of users who have their subset of bits B equal to a particular value \mathbf{v} . Suppose for a second that we are not concerned about the efficiency of our representation. How can we estimate this fraction while hiding the real values? Consider a

All Possible Private Values:	000	001	010	011	100	101	110	111
User Indicator Vector	0	0	0	0	1	0	0	0
User Published Vector	1	0	1	1	1	0	1	0

Figure 1: A very private (but very inefficient) publishing method. The user holds a private 3-bit value='100' which is first represented as $2^3 = 8$ -bit indicator vector (with a '1' in the position corresponding to '100'). This vector is then perturbed and published.

user \mathbf{u} and imagine that for each possible value \mathbf{v} he publishes a p -perturbed indicator whether the user's real value is equal to \mathbf{v} or not. In other words, for k bit long subset B , we represent the user's data by a 2^k bit long vector, which contains zeros everywhere, except at the position \mathbf{v} – the bit corresponding to the user's true value. At the end we perturb each bit with probability p and publish this perturbed vector. An example of this procedure is presented in the Figure 3.

Note that the published sequence is almost equiprobable to be generated from *any* possible underlying value, since for any possible user values \mathbf{v}' the original 2^k indicator vector differs in only 2 bits (corresponding to \mathbf{v}' and \mathbf{v}'').

Now, if we want to learn how often the value \mathbf{v} occurs in the database, we just look up the column corresponding to the value \mathbf{v} , and there for each user we have a perturbed indicator of whether the value is \mathbf{v} or 0. To obtain an estimate, we use the analysis for single bits discussed in section 2.

Evidently, publishing 2^k bits for each subset S is not very practical. Fortunately this can be avoided. Notice that the process used to generate these 2^k bits is extremely simple – all bits except one are generated using a p -biased coin, and the bit corresponding to the actual value is generated using a $(1 - p)$ -biased coin. This sequence can be simulated using *pseudorandom functions* which we only define intuitively. A precise formal definition is beyond the scope of this paper, however a very nice introduction to them can be found in [13].

To develop intuition, assume that we have an oracle which, when provided a key s , generates a random function f_s , such that for random *key* and for any value \mathbf{v} , $\Pr[f_s(\mathbf{v}) = 1] = p$ with all values being mutually independent. We emphasize here that each value is chosen only *once* for each key and each value. Thus for a fixed key s the function f_s is entirely deterministic. However before we evaluate $f_s(\mathbf{v})$ we have no way of knowing what the value is going to be. Suppose now, that a user holds value \mathbf{u} . Then, we want to modify the process of key selection so that the value of $f_s(\mathbf{u})$ would be more biased towards 1. E.g. if we just output random s then $f_s(\mathbf{u}) = 1$ with probability p , whereas we want it to be $1 - p$.

Intuitively, a user can seek out the necessary bias by skewing the distribution of keys. More specifically, a user selects a random key and then rejects it with non-zero probability if $f_s(\mathbf{u}) = 0$, and accepts it with probability 1 if $f_s(\mathbf{u}) = 1$. We call the key generated by this process a *sketch* (of a subset B). We emphasize here, that the outcome of this process is no longer a key chosen uniformly at random - but is skewed so that $f_s(\mathbf{u}) = 1$ is more likely.

Note that in order to keep independence, the oracle must generate functions independently for each user and each bit subset. To achieve this, we use a single random function which takes these values as input parameters. Indeed, suppose the function H takes as input a unique user identifier id , a bit subset B , a value \mathbf{v} and a key s (the one used as input to the oracle above). The value of such a function at a given input is chosen to be 1 or 0 using a p -biased coin. Different user identifiers will ensure that each user receives a random function that is independent of everybody else's function. Of course, we still need an efficient way to represent such a function. Fortunately there are standard algorithms for doing this. For example, any collision free secure hash (such as MD5 [21] or WHIRLPOOL [4]) is an example of such a function².

Observe that a function which returns a uniform value can be transformed to mimic a p -biased coin flips using a simple algorithm. Indeed, suppose we have a collision free function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, and we would like to obtain a binary function $H : \{0, 1\}^* \rightarrow \{0, 1\}$ such that for random x , $H(x) = 1$ with probability p . To achieve this, we write p in binary form, $p = \sum_{i=1}^t p_i 2^{-i}$. We assume that p can be written using only λ bits³. Then for a given input x let $\mathcal{H}(x) = v_1 \dots v_\lambda$. We report 1 if $\sum_{i=1}^\lambda v_i 2^{-i} \leq \sum_{i=1}^\lambda p_i 2^{-i}$

²It is also possible to generate a new function for each database individually using standard constructions of [13]

³Standard hash functions have length 128-512 bits, which is much larger than the typical precision used to represent real

and 0 otherwise. We now present our sketching algorithm.

Algorithm 1 THE SKETCHING ALGORITHM: $\text{Sketch}(id, \mathbf{d}, B)$

Input: Pseudorandom p -biased function H , security parameter p , user data (id, \mathbf{d}) , subset $B \subseteq [1..|\mathbf{d}|]$.

Output: A sketch s for \mathbf{d}_B .

- 1: Choose s uniformly at random without replacement.
 - 2: **if** $H(id, B, \mathbf{d}_B, s) = 1$ **then**
 - 3: Publish s and stop.
 - 4: **else**
 - 5: With probability $\frac{p^2}{(1-p)^2}$ publish s and stop. With probability $1 - \frac{p^2}{(1-p)^2}$ continue to step 1.
 - 6: **end if**
 - 7: If all values of s are exhausted then report *failure* and stop.
-

The Sketching Algorithm Note that the original randomized response is a special case of our technique where we sketch each bit individually. However the main advantage of our approach is in sketching large subsets of bits.

Analysis of the algorithm There are several questions that need to be addressed. First, we need to estimate the required length of a sketch and the running time of Algorithm 2. Second, we prove that the sketch preserves privacy. And the last question, that we defer until the next section, is what queries can be approximately answered with the sketch.

Let us begin with a more formal introduction to pseudorandom functions. Imagine the space of all functions that map n bit strings to n bit strings. Now imagine a uniform distribution over this space of functions – this is a truly random function ensemble. Pseudorandom function ensembles are ensembles that cannot be distinguished from truly random function ensembles by any efficient algorithm that can probe values of the functions at arguments it selects. Put another way, there is no algorithm that can distinguish between a function drawn from a pseudorandom ensemble and a truly random ensemble when given the ability to examine the function at various points.

While it is widely believed that pseudorandom functions do exist, their existence has not been proven. In our approach, as we show, the existence of pseudorandom functions is somewhat less crucial since privacy is unaffected by their existence, and furthermore it is unlikely that non-adversarially chosen queries will ever expose their non-randomness. Indeed, if such queries did exist, then we would be able to differentiate between a ‘currently-thought-to-be’ pseudorandom function and a random one – and that would be a breakthrough for modern cryptography. As for the current state of the art, if the length of the generator key is at least 300 bits⁴, it is unfeasible to build an algorithm whose answers on a pseudorandom function will differ from those it would produce on a truly random function. Thus without loss of generality we can assume that all values of our function H were chosen uniformly at random.

We assume that there is a public pseudorandom function H , which upon receiving a random binary string returns 1 with probability p and 0 otherwise. It is useful to think about a pseudorandom function as a black box such that for every set of parameters for which we have not yet evaluated our function, the value is generated randomly on the fly. Indeed, such an interpretation is possible since from the point of view of a user the values are computationally indistinguishable from those that are chosen at random. This substitution allows us to compute probability distributions and prove tail bounds over outcomes of pseudorandom functions even though the actual answer is of course entirely deterministic.

The first question we address is how many bits a user needs to represent a sketch so that Algorithm 1 fails with very small probability. It turns out that the number of bits we need is doubly logarithmic in the number users and the failure probability.

values. Furthermore if the need arises, the length of the output of a hash function can be increased [13].

⁴Here we are talking about the key used to define the global pseudorandom function for the entire database, not the short keys used to generate sketches

Lemma 3.1 (Minimal length of the sketch) *If the length of a sketch is at least $\ell = \lceil \log \frac{\log \frac{M}{\tau}}{\log 1-p^2} \rceil$, and there are at most M users in the system then probability that publishing algorithm fails for any user is less than τ .*

Proof. Consider a user with identifier id , a subset B , and suppose that the user's value on B is \mathbf{v} . For each key value we've considered, the algorithm stops with probability at least p^2 . Now, in order for our algorithm to fail, it should be the case that it did not stop on any of them. Thus the probability of failure is at most

$$(1-p^2)^{2^\ell} \leq (1-p^2)^{\lceil \frac{\log \frac{M}{\tau}}{\log 1-p^2} \rceil} \leq \frac{\tau}{M}.$$

where we use the fact that a ℓ -bit long binary value encodes 2^ℓ different values. Using the union bound, we get the desired result. ■

Note that the length of the sketch grows very slowly compared to the number of users and τ and grows *independently* of the length of the data we are sketching. For example if $p > 1/4$, then a 10 bit sketch is sufficient for any foreseeable practical use.

In terms of the algorithm's running time, note that the double logarithmic bound on the sketch length implies a logarithmic bound on the number of iterations. Since the algorithm only tries every sketch at most once, there will be at most $\frac{\log M/\tau}{\log(1-p^2)}$ iterations in the worst case. Furthermore it can be easily shown that since the algorithm terminates on each iteration with probability $\frac{p}{(1-p)^2}$ the expected running time is much less – and in fact is less than $\frac{(1-p)^2}{p^2}$ iterations.

In what follows, for clarity purposes all of our results will be conditioned on the fact that our algorithm does not fail. We start with a simple lemma which shows that the sketch published by Algorithm 1 indeed defines a function which is p -biased towards 1 on the user's real value, and p -biased towards 0 on all other values. The proof appears in Appendix D.

Lemma 3.2 (Correctness of the algorithm) *For a user (id, \mathbf{d}) , if the algorithm does not fail on a subset of bits B , it outputs a sketch s such that*

$$\Pr [H(id, B, \mathbf{d}_B, s) = 0] = p$$

and for all $\mathbf{v} \neq \mathbf{d}_B$,

$$\Pr [H(id, B, \mathbf{v}, s) = 0] = 1 - p,$$

where the probability is taken over all possible outcomes of the algorithm and all evaluations of H .

Now we prove our first privacy result that any sketch is almost equally likely to be generated from any private profile. As we mentioned before, our privacy guarantees are independent of the quality of the public pseudorandom generator, e.g. even an adversarial choice of the values of H would not compromise a user's privacy.

Lemma 3.3 (Privacy with a single sketch released) *For a subset B , if a user \mathbf{u} releases a sketch s according to Algorithm 1 then for any possible values of their profile \mathbf{d}' and \mathbf{d}'' we have*

$$\frac{\Pr [User publishes sketch s | User profile is $\mathbf{d}']}{\Pr [User publishes sketch s | User profile is $\mathbf{d}'']} \leq \left(\frac{1-p}{p}\right)^4$$$$

where the probability is taken only over the outcomes of the user's private coin flips and not over the outcomes of a public pseudorandom function.

Proof. Let $r = \left(\frac{p}{1-p}\right)^2$. Since we will only be considering a sketch of a single subset for B and for a single user with identifier id , we use $f(\mathbf{d}, s)$ as shorthand for $H(id, B, \mathbf{d}_B, s)$. We investigate how the algorithm runs given that the user profile is \mathbf{d} . In particular, we say that the key s evaluates to z (on user profile \mathbf{d}) if $f(\mathbf{d}, s) = z$. As in the previous result we assume that the sketch has length ℓ bits thus taking $\mathcal{L} = 2^\ell$ possible values.

We say that the algorithm *considers* a key s , if it samples s during its execution process. Let $Y_{\mathbf{d}s}$ denote the probability that the algorithm given a profile \mathbf{d} *considers* a key s . Note that if the key s evaluates to 1, then *considering* it is equivalent to publishing it. The probability $X_{\mathbf{d}s}$ that the user publishes key s is then bounded by

$$rY_{\mathbf{d}s} \leq X_{\mathbf{d}s} \leq Y_{\mathbf{d}s},$$

since the algorithm publishes a considered key with probability at least r , and it never publishes a non-considered key. Now we show that there exists Y_{\min} and Y_{\max} such that $Y_{\min}/Y_{\max} \geq r$ and for any possible user profile \mathbf{d} and s :

$$Y_{\min} \leq Y_{\mathbf{d}s} \leq Y_{\max}.$$

If we could show that then we have

$$rY_{\min} \leq \Pr[\text{User}(id, \mathbf{d}) \text{ publishes sketch } s] \leq Y_{\max} \leq Y_{\min}/r \quad (3)$$

independent of the actual value of \mathbf{d} , and the lemma would follow.

Let's prove those bounds. Notice that the behavior of the algorithm is invariant with respect to permutations of the key evaluations. Thus for any two profiles \mathbf{d}' and \mathbf{d}'' that have the same number of keys that evaluate to 1, we have $Y_{\mathbf{d}'s} = Y_{\mathbf{d}''s}$. Analogously, if for a given profile, two given keys evaluate to the same value then they have exactly the same probability of being considered. Denote by $Q(\mathbf{d})$ - a number of different keys s which evaluate to 1 on \mathbf{d} . Then we can write $Y_{\mathbf{d}s} = Z_{f(\mathbf{d},s)}^{Q(\mathbf{d})}$, where $Z_w^{(q)}$ denotes the probability of a particular key which evaluates to $w \in \{0, 1\}$ to be considered, in the case when exactly q keys evaluate to 1. Now we abstract from the underlying profile, and just concentrate on how the algorithm behaves as a function of how many keys evaluate to "1" or "0". This includes all possible profiles as well.

Since there are $\mathcal{L} = 2^\ell$ possible keys, the values of $Z_0^{(\mathcal{L})}$ and $Z_1^{(0)}$ are undefined, and for the remaining values $0 \leq q < 2^{\ell-1}$ we have $Z_0^{(q)} = Z_1^{(q+1)}$. Indeed, we *consider* every key before we evaluate its value, therefore the probability that the algorithm *considers* a key which evaluates to 0 is *equal* to the probability of considering a key which evaluates to 1 - where the rest of the evaluations stay the same (but an extra key has evaluated to 1). Thus it is sufficient to only consider $Z_1^{(q)}$. We are going to omit the lower index and use simply $Z^{(q)}$.

Now, if all keys evaluate to one, then the algorithm always terminates on the first iteration, therefore each key has equal probability of being considered. Thus $Z^{(\mathcal{L})} = \frac{1}{\mathcal{L}}$. It is also easy to see that $Z^{(q)} \geq Z^{(q+1)}$ since changing the evaluation of a single key to "0" increases the probability of having one more iteration (and hence each key which evaluates to "1" is more likely to be considered). Therefore $Y_{\max} = Z^{(1)}$ and $Y_{\min} = \frac{1}{\mathcal{L}}$. Now we only need to compute $Z^{(1)}$ - the probability that the algorithm chooses "1", if all hash values but one are evaluated to zero. Let's compute the probability V_i that the algorithm chooses the key which evaluates to "1" at the iteration i . We have:

$$V_i = \left[\prod_{j=0}^{i-1} \left(1 - \frac{1}{\mathcal{L}-j}\right)(1-r) \right] \times \frac{1}{\mathcal{L}-i} = \frac{(1-r)^i}{\mathcal{L}-i} \times \left[\frac{\mathcal{L}-i}{\mathcal{L}-i+1} \times \frac{\mathcal{L}-i+1}{\mathcal{L}-i+2} \times \dots \times \frac{\mathcal{L}-1}{\mathcal{L}} \right] = \frac{(1-r)^i}{\mathcal{L}}$$

where the first term is the probability that the algorithm does not terminate on the first $i-1$ iterations, and the second term is the probability that it chooses the key which evaluates to one. Therefore we have

$$Z^{(1)} = \sum_{i=0}^{\mathcal{L}} V_i = \frac{1}{\mathcal{L}} \sum_{i=0}^{\mathcal{L}} (1-r)^i \leq \frac{1}{r\mathcal{L}}$$

The result follows. ■

Corollary 3.4 (Privacy with many sketches) *If a user \mathbf{u} releases l sketches according to Algorithm 1 then for any possible values of their entire profile \mathbf{d}' and \mathbf{d}'' we have*

$$\left(\frac{1-p}{p}\right)^{4l} \leq \frac{\Pr[\text{User publishes sketches } s_1 \dots s_l \mid \text{User profile is } \mathbf{d}']}{\Pr[\text{User publishes sketches } s_1 \dots s_l \mid \text{User profile is } \mathbf{d}'']} \leq \left(\frac{p}{1-p}\right)^{4l}$$

In particular, if $p \geq 1/2 - \frac{\varepsilon}{16l}$, then

$$1 - \varepsilon \leq \frac{\Pr[s|\mathbf{d}']}{\Pr[s|\mathbf{d}'']} \leq 1 + \varepsilon$$

Proof. Since conditioned on a profile, each sketch is generated independently, the first part follows. The second part follows from the behavior of the exponent of the form $(1 + \varepsilon/q)^q \approx (1 + \varepsilon)$. ■

4 Utility

In this section we show that if we sketch a subset B , then we can answer an arbitrary query of the form $I(B, \mathbf{v})$ and bound the amount of noise introduced. Note that since we now operate with pseudorandom functions (instead of the truly random ones), the analysis of utility is slightly subtle. In particular, no result can be proven unless we assume the existence of pseudorandom functions. To simplify the exposition we prove our utility guarantees assuming that values of $H(\cdot)$ are chosen at random each time the function is computed on a new set of parameters. Then, assuming pseudorandomness of chosen $H(\cdot)$, we conclude that if the generating key of $H(\cdot)$ is long enough⁵, then the querying algorithm will produce the same result with probability only negligibly different from the one for a purely random function. (If not, then we have constructed an algorithm that can differentiate between a random and a pseudorandom function – which is unlikely.)

We begin by giving an algorithm that can be used to answer conjunctive queries over sketched subsets.

Algorithm 2 CONJUNCTIVE QUERY

Input: Pseudorandom function H , database of sketches $\mathcal{S}(id, B)$, subset of bits involved in the query B and querying value \mathbf{v}

Output: Approximate fraction of users who satisfy the query $\mathbf{d}_B = \mathbf{v}$

- 1: Compute fraction \tilde{r} of users who satisfy $H(id, B, \mathbf{v}, \mathcal{S}(id, B)) = 1$.
 - 2: Report $r' = \frac{\tilde{r}-p}{1-2p}$.
-

Now we show that this algorithm produces an answer which is not too far away from the true answer.

Lemma 4.1 (Quality guarantee for algorithm 2) *Assuming that H is a pseudorandom function, the answer r' produced by the algorithm 2, is different from the true answer r by more than ε with probability at most $\exp[-\frac{\varepsilon^2(1-2p)^2M}{4}]$.*

Equivalently, if p is bounded away from $1/2$, then for any δ with probability $1 - \delta$, the error introduced into an answer is at most $O(\sqrt{\frac{\log 1/\delta}{M}})$

Proof. By lemma 3.2 we have

$$E[\tilde{r}] = (1-p)r + p(1-r),$$

thus $r = \frac{E[\tilde{r}]-p}{1-2p}$, hence it is sufficient to show that with probability $\exp[-\frac{\varepsilon^2(1-2p)^2M}{4}]$, \tilde{r} differs from its expectation by a factor of $\varepsilon(1-2p)$. This immediately follows from the Chernoff inequality. Without loss of generality we can assume that $E[\tilde{r}] \geq \frac{1}{2}$ since we can always count the number of '0' entries instead. Thus, by the Chernoff inequality we have:

$$\Pr[|\tilde{r} - E[\tilde{r}]| \geq \gamma E[\tilde{r}]] \leq \exp[-\frac{\gamma^2 E[\tilde{r}] M}{2}] \leq \exp[-\frac{\gamma^2 M}{4}]$$

substituting $\gamma = \varepsilon(1-2p)$ we have the desired result. The second part follows. ■

⁵Here we are talking about the global key which defines the function for the entire database – not the keys that each user selects when publishing sketches. With the current state of the art 300 bit is more than sufficient

Given multiple sketches, we can also answer queries which involve unions of the subsets that those sketches describe. For example suppose that each user sketches subsets B_1, \dots, B_q . We can estimate how many users satisfy an arbitrary conjunctive query defined on the union $B = B_1 \cup \dots \cup B_q$ and a value $\mathbf{v} \in \{0, 1\}^{|B|}$.

This can be solved by solving a system of linear equations similar to the one introduced in [10]. The crucial difference from [10], is that now we can count frequencies of *large* itemsets - and these can be combined together using their technique for single bits. Due to space limitations, we overview this technique in Appendix F.

4.1 Computable queries

In this section we describe several types of queries which can be computed using only a constant number of conjunctive queries. Our description has an empirical character - we do not provide a formal notion of what kind of queries we can compute but rather show how to compute different queries. We believe that these empirical observations are an important first step towards understanding the full power of conjunctive queries. Also, in the appendix we present an example of a query which is not immediately expressible with a constant number of conjunctive queries, and yet is efficiently computable. Thus there is the potential that sketches support a richer query language.

Boolean queries As we already mentioned bit queries correspond to the easiest type of boolean query. For example, if each user holds boolean values $\mathbf{d}_1 \dots \mathbf{d}_t$ then the conjunctive query on bits $i_1 \dots i_k$ corresponds estimating the fraction of users that satisfy $\mathbf{d}_{i_1} \wedge \mathbf{d}_{i_2} \wedge \dots \wedge \mathbf{d}_{i_k}$ where each term is either \mathbf{d}_{i_l} or $\bar{\mathbf{d}}_{i_l}$. Additionally using the system of equations similar to the one in Appendix F, one can estimate the fraction of users that satisfy exactly l out of k bits in the query.

More generally, one can estimate the fraction of users that satisfy a given decision tree. Each path in the decision tree corresponds to a single conjunctive query and any user satisfies at most one path of the decision tree. Thus the total fraction of users who satisfy a decision tree is simply the sum of the fraction of users that satisfy each path (conjunctive query).

Computing Means/averages For the rest of this section, we assume that each profile holds several k -bit integer attributes a, b, c , etc, that are stored in binary form in the user's profile \mathbf{d} . The value of an attribute for user u is denoted by a_u . Let \mathbb{A} denote the subset of bits used to store the value of attribute a in the user profile. Furthermore, let \mathbb{A}_i denote the subset which contains the i highest bits of a . Let A_i denote the index of the i th highest bit. If user u has profile \mathbf{d} then $\mathbf{d}_{\mathbb{A}}$ is a_u written in binary notation and $\mathbf{d}_{\mathbb{A}_i}$ is the value of the i -th highest bit. Also to avoid multiple subscripts we will denote $\mathbf{d}_{\mathbb{A}_i}$ by a_{ui} .

We begin with the simplest type of query: computing the sum (or average) $S = \sum_{u \in U} a_u$. We expand the binary representation of a_u

$$a_u = \sum_{i=1}^{k-i} a_{ui} 2^{k-i}. \quad (4)$$

Then we have:

$$S = \sum_{u \in U} \sum_{i=0}^{k-1} a_{ui} 2^{k-i} = \sum_{i=1}^k 2^{k-i} \sum_{u \in U} a_{ui} = \sum_{i=1}^k 2^{k-i} I(A_i, 1)$$

Note that after we rearrange the sum order the inner sum becomes a single-bit query on bit A_i . Thus if each bit gets released, it is sufficient to release the sketch of each bit in the underlying binary representation.

Similarly we can compute the inner product of two non-integer attributes a and b :

$$S = \sum_{i \in U} a_i b_i = \sum_u \sum_i \sum_j 2^{2k-i-j} a_{ui} b_{uj} = \sum_i \sum_j 2^{2k-(i+j)} \sum_{u \in U} a_{ui} b_{uj} = \sum_i \sum_j 2^{2k-(i+j)} I(A_i \cup B_j, 11)$$

where the last transition follows from the fact that $a_{ui} b_{uj}$ is '1' if and only if both a_{ui} and b_{uj} are both '1'. Therefore $\sum_u a_{ui} b_{uj}$ corresponds to a bit query "how many users have their bits a_i and j set to one". That is, the inner product can be written as k^2 2-bit queries.⁶ Notice that here we do not have to sketch each

⁶The number of queries can be reduced if we ignore terms which contribute much less than the expected noise in the answer.

pair $A_i B_j$, but we can just use individual sketches and then use the technique from Section F to “glue” them together.

Interval queries We now consider queries of the form “how many users satisfy $a_u \leq c$ ” for constant c . In the case when a denotes the user’s salary, this corresponds to a query of the form “How many users have salary less than c ?”. As usual, we consider c in binary notation: $c = \overline{c_1 c_2 \dots c_k}$. How do we say that number $x = \overline{x_1 x_2 \dots x_k}$ is less or equal than c ? $x \leq c$ if and only if there exists i , such that for $j < i$ $x_j = c_j$ and $x_i < c_i$. For any x , at most one such i exists. Note that since x_i and c_i are either 0 or 1, in order to satisfy the last inequality we must have $x_i = 0; c_i = 1$. But then, for each i , this is a single bit query. Thus in order to count all possible $a_u \leq c$, for every i , such that $c_i = 1$, we need to pose a query about the first $k - i$ bits, such that $a_{u1} a_{u2} \dots a_{ui} = c_1 \dots c_{i-1} 0$, or reusing our I notation we have

$$|\{u : a_u \leq c\}| = \sum_{i=0}^{k-1} c_i I(\mathbb{A}_i, c_1 \dots c_{i-1} 0).$$

Notice that the number of queries we need to ask is equal to how many ‘1’s are in the binary representation of c , with the upper bound being the length of the integer.

However if we try to use a similar approach to answer the query $a_u + b_u < c$, it turns out that it requires exponential (in k) number of conjunctive queries. By using a slightly different approach, presented in Appendix E, we can express the query with a smaller number of conjunctive queries.

Combining queries together Suppose we are interested in estimating the number of users who for two constants c and d satisfy $a_u = c$ and $b_u < d$. The number of users who satisfy $a_u = c$, can be estimated by a single conjunctive query: $I(\mathbb{A}, c_1 \dots c_k)$. The number of users who satisfy the second constraint can be estimated by posing k bit queries $I(\mathbb{B}_i, d_1 \dots d_i)$, for each $1 \leq i \leq k$. Now, to compute the number of users that satisfy both conditions we just need to compute k queries of the form: $I(\mathbb{A} \cup \mathbb{B}_i, c_1 \dots c_k d_1 \dots d_i)$.

Analogously, we show how to compute the average value of b_u such that $a_u \leq c$:

$$\sum_{j=1, k, c_j=1}^k \sum_{i=1}^k 2^{k-i} I(\mathbb{A}_j \cup \mathbb{B}_i, c_1 \dots c_{j-1} 01)$$

Similarly one can combine constraints on different variables into queries about users who satisfy all those constraints.

5 Conclusions and open problems

We presented a technique that enables approximate computation of various queries on private user data which has very strong privacy guarantees. Our results are based on sketching - a novel technique that allows every user to publish a sketch of their data in such a way that it is impossible to gain significant confidence about the true value - even for an attacker with arbitrary knowledge. There are still many open questions. First, is it possible to formally describe what kind of queries can be formulated using only a limited number of conjunctive queries? An ideal solution would be a query language, such that any query in this language would require only linear, (or polynomial) in the length of the query, number of conjunctive queries.

Also a natural generalization of sketching bit subsets is sketching arbitrary functions of a user profile. The same privacy guarantees apply, but the main question is whether we can significantly expand the range of queries we can answer.

We presented a worst cases analysis of the privacy if the user publishes a fixed number of sketches. It would also be interesting to see if these results could be improved by taking into account independence between different parts of the data, and/or relaxing the privacy definition. In particular, if one is willing to relax privacy guarantees from deterministic to negligibly small probability of leak then the result of Theorem 3.4 can be improved to allow quadratically more sketches while giving essentially same privacy guarantees.

6 Acknowledgments

The authors would like to thank Kobbi Nissim for fruitful discussions.

References

- [1] G Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K Kenthapadi, Nina Mishra, R. Motwani, U. Srivastava, J. Widom D. Thomas, and Y. Xu. Enabling privacy for the paranoids (vision paper). In *Proc. of VLDB*, 2004.
- [2] G. Aggarwal, T. Feder, K.Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and An Zhu. Anonimizing tables. In *ICDT*, 2005.
- [3] Rakesh Agrawal, Ramakrishnan Srikant, and Dilys Thomas. Privacy preserving olap. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 251–262, New York, NY, USA, 2005. ACM Press.
- [4] P.S.L.M. Barreto and V. Rijmen. The whirlpool hashing function. In *NESSIE Workshop*, 2000.
- [5] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, New York, NY, USA, 2005. ACM Press.
- [6] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Theory of Cryptography Conference*, 2005.
- [7] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.
- [8] D. Dobkin, A. Jones, and R. Lipton. Secure databases: protection against user influence. *ACM Trans. Database Syst.*, 4(1):97–106, 1979.
- [9] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, 2004.
- [10] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222. ACM Press, 2003.
- [11] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agarwal, and Johannes Gehrke. Privacy preserving mining of association rules. *Inf. Syst.*, 29(4):343–364, 2004.
- [12] O. Goldreich. *Foundations of Cryptography, Basic Tools*. Cambridge University Press, 2001.
- [13] O. Goldreich. *Foundations of Cryptography, Volume II*. Cambridge University Press, 2004.
- [14] <http://www.macworld.com>. Hackers breach lexisnexis, graph info on 32,000 people. 2005.
- [15] <http://www.msnbc.msn.com>. Data theft affects 145,000 nationwide. suspect arrested in choicepoint case agrees to plea deal. 2005.
- [16] <http://www.washingtonpost.com>. Northwest airlines faces privacy suits. 2004.
- [17] <http://www.wired.com>. Acxiom opts out of opt-out. 2003.
- [18] Krishnaram Kenthapadi, Nina Mishra, and Kobbi Nissim. Simulatable auditing. In *Proc. of PODS*, pages 118–127, New York, NY, USA, 2005. ACM Press.
- [19] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. *Journal of Computer and System Sciences*, 6:244–253, 2003.

- [20] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian. l-diversity: Privacy beyond k-anonymity. In *Proc. of ICDE*, 2006.
- [21] Ronald Rivest. The md5 message-digest algorithm. In *RFC 1321*. MIT LCS & RSA Data Security Inc, 1992.
- [22] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, 1998.
- [23] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. on Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.
- [24] S. Warner. Randomized response: A survey technique for eliminating error answer bias. *J. of American Statistical Association*, 1965.

A Trusted third party

In this section, we give some useful insight into how our input perturbation technique can be used to solve the output perturbation problem with better privacy guarantees, albeit less utility.

We consider a system that contains a database with private information and that answers queries about the content of this database. The goal is to perturb answers in such a way that it is impossible to gather any personal information about any individual user data stored in the database. To use our scheme, the system administrator devises the set of bit subsets that need to be sketched and computes the sketches on each row of the database. Then, for each query, the system computes the answer using only the sketches (and not the actual user data). The only thing the attacker can potentially learn are the sketches themselves. And we already know that the sketches do not leak information about the user’s real data.⁷

With this simple approach, we can overcome a negative result of Dinur and Nissim[7] and Blum et al [5] which suggests that linear noise must be added in order to protect from an attacker with unlimited computational power. In our case we only add⁸ $O(\sqrt{M})$ in all except a negligible fraction of the queries. So while technically in the worst case, the added noise could be linear, the chance that the worst case happens decrease exponentially as the size of the database increases. For example, in a system with M users, the chance we encounter such a bad query is $2^{-\omega(M)}$. This result is tight in the following sense. It is impossible to devise a system that would have noise smaller than $O(\sqrt{M})$ in all but a negligibly small number of queries. This follows from one of the results of the same paper[7].

From a practical point of view, one might want to implement both input and output perturbation in their system, and then offer two types of access (for example paid and free). The paid mode would correspond to output perturbation (for example SULQ framework [5]) and would only add a small noise $E \leq \sqrt{M}$ to the system. However, the total number of queries answered in this mode is limited (by the minimum of E^2 and the total number of users in the database). Once the limit of queries is exhausted the system will stop answering those queries. Even before the system exhausts paid queries, it can be used in the second mode, where it adds noise $O(\sqrt{M})$, but the database can answer an unlimited number of queries. Note that the amount of noise that the system adds is about the same, as SULQ adds in the situation where it is tuned to answer as many queries as possible. In a way, our scheme closes the gap between how much noise should be added in the case when we only allow a finite or infinite number of queries.

B Single Bit Flipping is ϵ -private

Lemma B.1 *In a single column database, the bit-flipping procedure is ϵ -private provided that $p = 1/2 - c\epsilon$ and $c \leq 1/4$.*

⁷Even learning the values of sketches is challenging in the case of a trusted party since the only information the attacker can infer explicitly is the evaluations of the pseudorandom function. This fact simplifies the proof of lemma 3.3 and slightly improves the privacy guarantee.

⁸Here and elsewhere in this section we follow [5] and assume that each user contributes at most 1 to the answer - of course the result could be scaled to any arbitrary value

Proof. For $v = 1$,

$$\frac{\Pr(\tilde{x}_i = 1|x_i = 0)}{\Pr(\tilde{x}_i = 1|x_i = 1)} = \frac{p}{1-p}$$

and for $v = 0$,

$$\frac{\Pr(\tilde{x}_i = 0|x_i = 0)}{\Pr(\tilde{x}_i = 0|x_i = 1)} = \frac{1-p}{p}$$

Thus it suffices to determine for what p , we simultaneously have that

$$\frac{p}{1-p} \leq 1 + \epsilon$$

and

$$\frac{1-p}{p} \leq 1 + \epsilon$$

For $p = 1/2 - c\epsilon$, it can be shown that the bit flipping procedure is ϵ -private provided that $c \leq 1/4$. ■

C Different privacy definitions

The definition of privacy that is most similar to ours is (ϵ, δ, T) -privacy introduced in [7, 5]. Essentially, a perturbation satisfies this definition if each user is guaranteed to have ϵ -privacy with probability at least $1 - \delta$ given that the attacker answers at most T queries. ϵ -privacy is equivalent to $(\epsilon, 0, \infty)$ -privacy of [5]. Note that even though ϵ -privacy is a stronger definition, the (ϵ, δ, T) -privacy was used in a much less general context than ours (that is, the third party answers queries and data is never released). But the benefit is that it answered a much richer set of queries .

Other definitions include ρ_1 -to- ρ_2 privacy breach [10, 11] which occurs when the prior probability of any predicate of a user's data \mathbf{d} is at most ρ_1 while the posterior probability of $Q(\mathbf{d})$ given sanitized information s is at least ρ_2 . Typical values for ρ_1 and ρ_2 are in the range 10 – 90%. It can be shown [10] that ϵ -privacy implies ρ_1 -to- ρ_2 privacy, but not vice versa. In fact, ϵ -privacy is a much stronger definition. In particular, it bounds the relative change of the posterior to prior probability, but not the absolute change. For example, let $\rho_2 = 50\%$ and let the prior probability that a user is HIV+ be 0.001%. Then if an attacker learns that the posterior probability that a user is HIV+ is 49%, it is not considered a privacy breach – even though the attacker learned an enormous amount about the user.

We emphasize here that this problem is not alleviated by any single choice of ρ_1 and ρ_2 . Note, however, that the perturbation scheme described in [10, 11] does actually satisfy our privacy definition.

A similar but even weaker definition called (s, ρ_1, ρ_2) -privacy is suggested in [3]. In particular ρ_1 to ρ_2 -privacy implies (s, ρ_1, ρ_2) -privacy, but not vice versa. Furthermore in contrast with [10], the method described in [3] can not be extended to our stronger privacy definition. The main idea in [3] is to keep each attribute with relatively high probability, and replace it with noise otherwise. Unfortunately, this method is susceptible to partial knowledge attack as explained in the introduction.

In the case when the database is perturbed by a third party, there are other notions of privacy, such as k -anonymity where the guarantee is that each user profile is indistinguishable from at least $k - 1$ others. While this is a very nice and appealing framework – since no data is changed, only suppressed – it has been shown that it might lead to privacy breaches in some applications. For more details we refer to Machanavaajhala et al [20] where an interesting notion of l -diversity is introduced.

D Proof of Lemma 3.2

Proof of Lemma 3.2 To prove the second part it is sufficient to notice that our choice of s is independent from any $B' \neq \mathbf{d}_B$, thus $\Pr[H(id, B, \mathbf{v}, s) = 1] = p$. For the first part, let T_t denote the event that the algorithm terminates on iteration t . Since the algorithm did not fail, $\sum_{i=1}^{2^\ell} \Pr[T_i] = 1$. Recall that we sample our key values without replacement, thus on each cycle H is evaluated on different values. Therefore, we can use the trick where all values of H are assumed to be generated on the fly. Consider what happens

at timestep t . Let s_t denote our key value at iteration t (if we have reached it). Recall that if $f(s_t) = 1$ then the algorithm deterministically terminates, and if $f(s_t) = 0$ it terminates with probability $\frac{p^2}{(1-p)^2}$. Therefore $\Pr [T_t] = p + \frac{p^2}{1-p}$. Hence we have:

$$\Pr [f(s_t) = 1|T_t] = \frac{p}{p + \frac{p^2}{1-p}} = 1 - p, \text{ and } \Pr [f(s_t) = 0|T_t] = \frac{\frac{p^2}{1-p}}{p + \frac{p^2}{1-p}} = p.$$

Therefore

$$\Pr [\text{algorithm publishes } s \text{ s.t. } f(s) = 1] = \sum_{t=1}^{2^\ell} (\Pr [f(s_t) = 1|T_t] \Pr [T_t]) = (1 - p) \sum_{t=1}^{2^\ell} \Pr [T_t] = 1 - p$$

■

E Example of a query not directly expressible as a few bit queries

In this section we present an example of a query that cannot be easily translated into a small number of conjunctive queries. Yet via variable substitution we can efficiently answer the query. For the purpose of this section, we assume that each bit of the database is simply p -perturbed – or equivalently we sketch every single bit.

We wish to determine “How many users satisfy: $a_u + b_u < 2^r$?” where we assume that $a_u = \overline{a_{u1}a_{u2} \dots a_{uk}}$ and $b_u = \overline{b_{u1}b_{u2} \dots b_{uk}}$ are k -bit integers. As per our assumption, we have \tilde{a}_{ui} and \tilde{b}_{ui} which are p -perturbations of each bit of a_u and b_u respectively. In order to satisfy the constraint we must have

$$a_{ui} = b_{ui} = 0, \text{ for all } i \geq r.$$

Now, if $a_{u(r-1)} = b_{u(r-1)} = 0$, then the constraint is automatically satisfied - no matter what the other bits are $a_u + b_u < 2^r$. If on the contrary $a_{u(r-1)} = b_{u(r-1)} = 1$ then the constraint is automatically violated. Finally if $a_{u(r-1)} + b_{u(r-1)} = 1$, then we have to check the $r - 2$ bit, where the same rule applies. Thus, for each $j \leq r - 1$, we need to compute how many users satisfy:

$$a_{ui} + b_{ui} = 1, \text{ for } i > j \text{ and } a_{uj} = b_{uj} = 0; \tag{5}$$

So we need to ask all bit queries where for each i , exactly one a_{ui} and b_{ui} is '1' and another is '0' – thus asking an exponential number of them. To avoid this, we introduce a binary variable $q_{ui} = [(a_{ui} + b_{ui}) = 1]$. Given a p -perturbed version \tilde{a}_{ui} and \tilde{b}_{ui} , it is easy to see that $\tilde{q}_{ui} = \tilde{a}_{ui} \oplus \tilde{b}_{ui}$ are $2p(1-p)$ -perturbed variants of q_{ui} - since the evaluation changes if and only if exactly one of a_{ui} and b_{ui} get perturbed. But now we can use all our machinery on the “virtual” bits as well. In particular we can compute how many users satisfy $q_{ui} = 1$ for all $i > j$. Thus for any i , the number of users satisfying (5) can now be efficiently computed.

With some extra effort the query can be generalized to $a_u + b_u < c$, where c is arbitrary constant, but we omit that discussion.

F Combining sketches together

In this section we describe how, given sketches for subsets $B_1 \dots B_q$, we can estimate how many users satisfy an arbitrary conjunctive query defined on the union $B = B_1 \cup \dots \cup B_q$ and the value $\mathbf{v} \in \{0, 1\}^{|B|}$. Let $\mathbf{v}_1 \dots \mathbf{v}_q$ denote the projection of \mathbf{v} into subsets $B_1 \dots B_q$. Let a sketch for user u of a subset i be denoted s_{ui} . Then for each user (id, \mathbf{d}) and index i , we have a perturbed virtual bit (as defined by a function $H(id, B, \mathbf{v}_i, s_{ui})$) indicating whether their true profile matches the query (B_i, \mathbf{v}_i) . Thus, to answer the query about (B, \mathbf{v}) we just need to estimate how many users have all unperturbed bits equal to “1”. Analogously, by estimating how many users have these bits equal to “0”, we learn how many users do not satisfy any query of the form $I(\mathbf{v}_i, B_i)$ – which could be used to estimate how many users satisfy a disjunction of conjunctions.

To simplify our exposition, we now abstract from sketches. Instead, we solve the following problem. Given k bits from each user, where each is changed with probability p , estimate how many users originally have all k bits set to one. This would solve the problem with sketches (where we use the values of the pseudorandom function as perturbed bits).

Without loss of generality assume that the user profile \mathbf{d} has k bits. Now, each original value \mathbf{d} , has a fixed probability of being perturbed into any other profile $\tilde{\mathbf{d}}$.

$$w[\mathbf{d} \rightarrow \tilde{\mathbf{d}}] = p^{\|\mathbf{d}-\tilde{\mathbf{d}}\|_1} (1-p)^{k-\|\mathbf{d}-\tilde{\mathbf{d}}\|_1}$$

where $\|\mathbf{d}-\tilde{\mathbf{d}}\|_1$ is the standard Hamming distance between boolean vectors. Since we observe the frequencies of all perturbed profiles, at least theoretically we can write a system of linear equations to solve for the frequencies of the actual values. But the system has size 2^k and is not feasible to solve in most cases. A similar approach was employed in Agrawal et al [3]. There however they considered non-binary data and justifiably argued that in most applications k is very small.

However, in our case, a better solution exists⁹. Since each bit is perturbed with the same probability, the system is very homogenous and one can reduce the size of the system of equations from 2^k to k .

Suppose B is a subset of bits, and without loss of generality suppose we are interested in how many users have all these bits equal to one (since our perturbation is symmetric, we can immediately generalize it to any other value). Let $v[l \rightarrow l']$ denote the probability that users who originally had l bits in B set to 1 will have l' bits set to 1, after each bit was perturbed. We evaluate all the different ways that we can obtain l' bits set to one if we originally had l bits set to one. If we switch h bits which were set to 1, then in order to obtain l' bits in the resulting value, we need to switch $l' - (l - h)$ bits which were originally set to 0. In order to avoid switching negative number of bits we must have $\max(0, l - l') \leq h \leq \min(l, k - l')$. For each h , there are

$$\binom{l}{h} \binom{k-l}{l'-l+h}$$

different ways to achieve this, and the probability of each choice happening is

$$p^h p^{l'-l+h} (1-p)^{k-(l'-l+2h)} = p^{l'-l} (1-p)^{k-(l'-l)} \left(\frac{p}{1-p}\right)^{2h}$$

Thus we have:

$$v[l \rightarrow l'] = p^{l'-l} (1-p)^{k-(l'-l)} \sum_{h=\max(l-l',0)}^{\min(k-l',l')} \left(\frac{p}{1-p}\right)^{2h} \binom{l}{h} \binom{k-l}{l'-l+h}, \quad (6)$$

Let x_l denote the number of users whose original profile matches l bits. We are interested in estimating x_k . Similarly let y_l' denote the fraction of users whose perturbed profile matched l' bits. Then we have

$$E[y] = Vx$$

where $V_{l'l} = v[l \rightarrow l']$, and y, x denotes a vector comprised of $y_0 \dots y_l$ and $x_0 \dots x_l$ respectively. Then we have:

$$x = V^{-1}E[y]$$

and in particular $x_k = V_l^{-1}E[y]$. $E[y]$ is hidden, however we can use an observed value of y as an approximation for $E[y]$. If the condition number of V is a constant C , and the system has ν users, then with high probability the error of estimation of x would be $O(\frac{C}{\sqrt{\nu}})$. However, an empirical analysis of the conditioning number of the matrix V , suggests that it decreases exponentially in k , with the base of the exponent proportional to $1/(p-1/2)$.

⁹Similar solution was proposed by Evfimievski et al [10], for slightly difference perturbation scheme