Functionally Private Approximations of Negligibly-Biased Estimators

André Madeira[1*], **S.Muthukrishnan**[1,2]

[1] Rutgers University
Piscataway, NJ, USA
{amadeira,muthu}@cs.rutgers.edu

[2] Google Research
New York, NY, USA
muthu@google.com

ABSTRACT. We study functionally private approximations. An approximation function g is *functionally private* with respect to f if, for any input x, $g(x)$ reveals no more information about x than $f(x)$. Our main result states that a function f admits an efficiently-computable functionally private approximation g if there exists an efficiently-computable and negligibly-biased estimator for f. Contrary to previous generic results, our theorem is more general and has a wider application reach.
We provide two distinct applications of the above result to demonstrate its flexibility. In the data stream model, we provide a functionally private approximation to the L_p-norm estimation problem, a quintessential application in streaming, using only polylogarithmic space in the input size. The privacy guarantees rely on the use of pseudo-random *functions* (PRF) (a stronger cryptographic notion than pseudo-random generators) of which can be based on common cryptographic assumptions. The application of PRFs in this context appears to be novel and we expect other results to follow suit. Moreover, this is the first known functionally private streaming result for *any* problem.
Our second application result states that every problem in some subclasses of ♯P of hard counting problems admit efficient and functionally private approximation protocols. This result is based on a functionally private approximation for the ♯DNF problem (or estimating the number of satisfiable truth assignments to a Boolean formula in disjunctive normal form), which is an application of our main theorem and previously known results.

1 Introduction

Consider a two-party functionality $f(x_1, x_2) = (y_1, y_2)$, where (x_i, y_i) is the private input/output pair of party $i \in \{1, 2\}$. Informally, a *private computation* of f is one that computes f correctly and guarantees that each party i learns only y_i and nothing else.

Interestingly, Feigenbaum et al. [1] observed that the private computation of an approximation function $g(x_1, x_2) = (\tilde{y}_1, \tilde{y}_2)$ of f can potentially leak more information than the computation of f itself. Indeed, consider function $f(x_1, x_2)$ computing the Hamming distance between binary vectors x_1 and x_2. Let g be an approximation of f where the least significant bit of $g(x_1, x_2)$ corresponds to some arbitrary bit of x_1 and all the remaining bits of g equals those of f. Although g is indeed a good approximation, it leaks more information about x_1 than f does. In view of this problem, the authors argued that it is natural to

[*] Supported in part by NSF grant CCF-0728937.

Foundations of Software Technology and Theoretical Computer Science (Kanpur) 2009.
Editors: Ravi Kannan and K. Narayan Kumar; pp 323–334
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.
Digital Object Identifier: 10.4230/LIPIcs.FSTTCS.2009.2329

require that g be also *functionally private* with respect to f; i.e. roughly speaking, there should be no (or it is computationally infeasible to find an) i such that $\tilde{y}_i$ "leaks" more information than y_i does (we make it precise in Section 2). As approximations are often used in place of exact computations to reduce computing resources, the definition also captures the notion that efficiency and privacy should not be conflicting goals.

We observe that although a series of seminal results [5, 19] claim that *any* efficiently-computable (read polynomial-time) distributed protocol for a functionality f can be "compiled" into a *private protocol*, one cannot claim the same for approximations. Indeed, the functional privacy property is inherent to the *description* of g and not of any protocol computing g. Hence, there is no hope for a "compiler-like" solution for approximations. Consequently, the focus on *functional privacy* has been on designing protocols for a particular set of functions of interest (or classes thereof). Unfortunately, since the definition of functional privacy first appeared in [1] few results have surfaced. Most are either tailored protocols for specific functions of interest [1, 2, 8, 10] or impossibility results [6]. An exception are the more general feasibility results of [1] that claims functionally private approximations for a specific set of conforming Monte-Carlo simulations. Unfortunately, the results are limited in scope and rigid in their requirements as we outline and discuss in Section 3.

Our main result, on the other hand, roughly states that a function f admits an efficient functionally private approximation g if there exists an efficient *negligibly biased* estimator for f. The result is flexible enough under many circumstances. We demonstrate this point by providing two distinct applications of it. The first relates to a quintessential problem in the data stream model of computation [12]: the estimation of the L_p norm of vectors, which in the non-private streaming setting spurred several new results. The second is concerned with feasibility results for $\sharp\mathsf{P}$ problems. Before presenting our contributions, we start with some relevant context.

Private Streaming Computations. Consider two parties Alice and Bob. Alice sees an n-dimensional vector a given as a series of coordinate updates. The j^{th} update is (j, j_i, j_u) where $j_i \in [n]$ refers to the dimension of the vector, and j_u the change to that dimension, i.e., $a[j_i] \rightarrow a[j_i] + j_u$. We visualize a as the stream. Each update has to be processed quickly and there is only limited memory to store a. Formally, we are allowed space polylogarithmic in n and various parameters of interest, as well as similar update and processing time. Similarly, Bob is given input vector b given as a stream. When a function f needs to be computed at time t, Alice and Bob communicate with each other to evaluate $f(a_t, b_t)$ where a_t (b_t) denotes Alice's (resp. Bob's) vector at time t (hereafter, we drop the subscript t whenever the context allows). Total communication is in bits polylogarithmic in n and other parameters. This is the distributed data stream model [12].

Our focus is on achieving *functionally private* protocols in the streaming model. In this setting, as in general private computation, Alice and Bob do not wish to reveal the contents of their streaming data. This stringent requirement is a result of either binding legal reasons or sheer competitiveness. However, in the spirit of cooperation or as required by law, they might be willing to perform a specific data analysis task in a secure way. This is the context for the problems we study. For the purposes of this paper, we will address a common streaming analysis that is already well-studied in the literature [7, 11, 14] (but in a secure way) and not delve deeper into its many applications (which can be found in [12]). Specifically, we consider the following problem: compute the L_p norm of vector $a - b$, denoted $L_p(a - b) = ||a - b||_p$, for $p \in [0, 2]$. Recall that $L_p(x) = (\sum_i |x_i|^p)^{1/p}$. Nearly all non-trivial streaming analyses — including the problem above — are in fact approximate (exact computations are impossible without linear space [12]) and hence we focus on *functionally private* approximations.

Private Computation of $\sharp$P-complete Problems. In this setting, Alice and Bob hold finite inputs a and b respectively. Similar in spirit as before, they wish to compute a $\sharp$P-complete function f of their private inputs such that no information other than $f(a, b)$ (and whatever can be inferred from it) "leaks". However, as $f(a, b)$ is an intractable problem, they must settle on computing an efficiently-computable functionally private approximation instead.

Results. Our contributions are as follows:

1. We show that if there exists a *negligibly biased* estimator (NBE) $\mathcal{A}(x, \epsilon', \delta')$ of $f(x)^\dagger$, which $\langle \epsilon', \delta' \rangle$-approximates‡ f for $\epsilon' = 1/2$ and $\delta' = \mu(\kappa)$ in time $\mathsf{poly}(\kappa, \log |x|)^\S$, and a public upper bound τ on $f(x)$, then there exists a *functionally private* $\langle \epsilon, \delta \rangle$-approximation g of f computable in time $\mathsf{poly}(\kappa, \log |x|, \log \tau, 1/\epsilon, \log(1/\delta))$ for a security parameter κ. Thus, if $\tau = \mathsf{poly}(|x|)$ as below, g is $\mathsf{polylog}(|x|)$-computable.
   The proof consists of taking enough samples from Bernoulli random variable (r.v.) with success probability $p = \mathcal{O}(\mathcal{A}(\cdot)/\tau)$ and ensuring $p = \Theta(1/c) \leq 1$ for a tight approximation using $\tilde{\mathcal{O}}(c)$ samples.¶ The output then depends solely on $\mathsf{E}[\mathcal{A}(\cdot)/\tau]$. Since this is negligibly far from $f(x)/\tau$ we argue that functional privacy is implied.
   This is a *general* result for any function f and is not limited to any format as opposed to the feasibility results in [1]. We believe that it is of general interest and will prove useful to other functionally private protocols such as the following results.
2. We design a *functionally private* $\langle \epsilon, \delta \rangle$-approximation g for the L_p norm, $p \in (0, 2]$, of an n-dimensional vector using $\tilde{\mathcal{O}}\left(\kappa^2 \log^2 n\right)$ bits of space on a security parameter κ.
   Our result is based on a slight adaptation of the recent non-private unbiased estimator for L_p [11] applied to our first result. To ensure functional privacy, we use a Pseudo-Random Functions (PRF), a stronger cryptographic notion than a Pseudo-Random Generator (PRG) that suffices for standard non-private streaming computations. Sampling from sketches and the use of PRFs in this context appear to be novel.
   From above, private streaming protocols for the L_p distance of two vectors follows. These are the first known private streaming protocols for any problem.
3. We design a *functionally private* $\langle \epsilon, \delta \rangle$-approximation g for the $\sharp$DNF problem, or estimating the number of satisfiable assignments of a formula in disjunctive normal form, a $\sharp$P-complete problem. In a nutshell, we rely on the result of Karp and Luby [9] to construct an unbiased estimator suitable for application of our first result.
   The result yields functionally private $\langle \epsilon, \delta \rangle$-approximations to all problems within some logic-based subclasses of $\sharp$P. Specifically, we show that $\sharp$DNF is complete under a private and approximation-preserving reduction for the $\sharp\Sigma_1$ and $\sharp\mathsf{R}\Sigma_2$ classes,

[†] informally, X is a NGE if $\mathsf{E}[X]$ is negligibly far from $f(x)$ and has finite variance. See Section 2 for details.

[‡] a function g $\langle \epsilon, \delta \rangle$-approximates f if $\mathsf{Pr}\left[|g(x) - f(x)| > \epsilon f(x)\right] \leq \delta$ for all inputs x.

[§] $\mathsf{poly}(n)$ ($\mathsf{polylog}$) means any polynomial in n (in $\log n$ respectively).

[¶] the notation $\tilde{\mathcal{O}}(n)$ should be read as $\mathcal{O}\left(n \log(1/\delta)/\epsilon^2\right)$ throughout the paper.

yielding functionally private approximations to all problems therein.

Although our goal is on achieving private protocols, we omit the details about constructing a secure two-party protocol. As Feigenbaum et al. [1] indicated, the challenge typically boils down to proving functional privacy when designing a private approximation protocol. Additionally, most of the construction details of a secure protocol are orthogonal to our main contributions in this paper. We refer the reader to [1, 3] for such details.

2 Preliminaries

Let $[m]$ denote the integer range $1, \dots, m$. We denote a *negligible* function in a positive integer parameter κ by $\mu(\kappa) \in \kappa^{-w(1)}$. A function f is said to be *overwhelming* if $1 - f$ is negligible. Polynomial time means time polynomial in n, $1/\epsilon$, and security parameter κ and is denoted by poly. Similarly, by $\mathsf{polylog}$, we mean time polylogarithmic in n, but poly in $1/\epsilon$ and κ. Finally, we say a function is *efficient* if it is poly-time computable.

DEFINITION 1.*[$\langle \epsilon, \delta \rangle$-approximation] A function g is an $\langle \epsilon, \delta \rangle$-approximation of f if, $\forall x$, $\Pr[|g(x) - f(x)| > \epsilon f(x)] \leq \delta$ holds for arbitrary $\epsilon, \delta \in (0, 1)$. The function g depends on both ϵ and δ and the probabilistic guarantees are over the randomness of g.*

Below is the general notion of indistinguishability of distributions in Cryptography.

DEFINITION 2.*[indistinguishability of distributions] Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are said to be* computationally *indistinguishable, denoted $\mathcal{D}_1 \overset{c}{\equiv} \mathcal{D}_2$, if for every pair of random variables $X_1 \sim \mathcal{D}_1$ and $X_2 \sim \mathcal{D}_2$ and for any family of polynomial-size circuits $\{C_\kappa\}$ we have $|\Pr(C_\kappa(X_1) = 1) - \Pr(C_\kappa(X_2) = 1)| \leq \mu(\kappa))$ for a security parameter κ. Distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are* statistically *indistinguishable, denoted $\mathcal{D}_1 \overset{s}{\equiv} \mathcal{D}_2$, if for any $X_1 \sim \mathcal{D}_1$ and $X_2 \sim \mathcal{D}_2$ the statistical distance $SD(X_1, X_2) = \frac{1}{2} \sum_a |\mathsf{Pr}[X_1 = a] - \mathsf{Pr}[X_2 = a]| \leq \mu(\kappa)$. Note that $\mathcal{D}_1 \overset{s}{\equiv} \mathcal{D}_2$ implies $\mathcal{D}_1 \overset{c}{\equiv} \mathcal{D}_2$ but not necessarily vice-versa.*

Consider the *functional privacy* definition for general approximations from [1].

DEFINITION 3.*[functional privacy [1]] A function g is* functionally private *with respect to a function f if there exists a probabilistic poly-time algorithm (a.k.a. simulator) $\mathcal{S}$ such that, for any input x, $\{\mathcal{S}(f(x))\} \overset{\tau}{\equiv} \{g(x)\}$ where $\overset{\tau}{\equiv}$ denotes either $\equiv$, $\overset{c}{\equiv}$, or $\overset{s}{\equiv}$.*

This definition captures the notion that the approximation output $g(x)$ does not reveal extra information about x besides what can be inferred from $f(x)$. Moreover, the functional privacy definition is independent of how g is computed or whether f is efficiently computable or not. Indeed, f could be a hard problem and thus $\mathcal{S}$ is modeled as having only access to $f(x)$ and not an oracle access to f.

3 Functional Privacy: current techniques and limitations

The seminal work of [1] presented a feasibility result for the following set of functions. Consider a two-party computation where Alice and Bob hold private inputs a and b of size n respectively and let x represent the input pair (a, b). Let $f(x) = \psi(\mathsf{Pr}[\xi])$, where ξ is an event or Bernoulli trial parameterized by a and b and ψ is an approximation-preserving function that is efficient to compute and invert. It was shown that f admits an efficient *functionally private approximation* g as long as $\mathsf{Pr}[\xi] \geq 1/\mathsf{poly}$. Essentially, g is constructed by applying ψ to the outcome of a sampling algorithm estimating $\mathsf{Pr}[\xi]$ directly from a and b via poly independent samples. Correctness follows from Chernoff bounds. On the other hand, the functional privacy simulator works as follows: given $f(x)$, apply ψ to poly independent samples of a Bernoulli random variable with success probability equal to an $\Omega(\kappa)$-bit approximation of $\mathsf{Pr}[\xi] = \psi^{-1}(f(x))$. Functional privacy follows from the fact that the simulated distribution is statistically indistinguishable (in a security parameter κ) from the one induced by g —and thus also computationally indistinguishable. Additionally, [1] extended the results to functions of the form $f(x) = \psi(\phi(\xi_1, \xi_2, \ldots, \xi_t))$ for a polynomial-size, constant-depth arithmetic formula $\phi(\cdot)$ of "coin manipulation" gates.[‖]

We outline some problems with the above feasibility results. The main drawback is the stringent structure on $f(x) = \psi(\phi(\cdot))$. It restricts f to be the result of some Monte-Carlo experiment, where coin manipulations suffices in making $\phi(\cdot)$ simulatable from $f(x)$ alone using $\psi^{-1}(\cdot)$. Unfortunately, this structure might not always be easily attainable. Indeed, for the problem we consider in Section 5, an efficient (and *known*) solution is to construct a coin $\phi(\cdot) = f(x)/h(x)$ for a function $h(x)$ not inferred from $f(x)$ alone. It turns out that $h(x)$ depends on the *structure* of x and thus of private inputs a and b. In that case, $\psi^{-1}(f(x))$ cannot yield $f(x)/h(x)$ properly as required without the knowledge of $h(x)$.

A second drawback is the requirement that $\mathsf{Pr}[\xi] \geq 1/\mathsf{poly}$. Essentially, it requires taking poly samples for a tight approximation. This might be prohibitive for very large inputs. In many cases, the only acceptable goal is to take $\mathsf{polylog}$ samples, as the sampling complexity is closely related to the communication complexity of a private distributed protocol [1]. Specifically, when a tighter range for $\mathsf{Pr}[\xi]$ is known, it is reasonable to expect a much better sampling complexity. Indeed, that is the case of the stand-alone private protocol of [8], which reduces the sampling complexity to $\mathsf{poly}(\kappa, \log n)$ by ensuring that $\mathsf{Pr}[\xi] \in \Theta(1/\kappa)$.

We address both concerns simultaneously. Roughly speaking, we show that it suffices to design a *negligibly biased* estimator (NBE) that $\langle \epsilon, \delta \rangle$-approximates f for f to admit a *functionally private approximation* g. Contrary to above, the NBE carries no restriction. For example, the NBE can be constructed out of a Monte-Carlo experiment or in any other way. In other words, it is applicable to *any* function f as long as a suitable NBE is available. Therefore, our result widens and also encompasses the previous feasibility results of [1].

3.1 Randomness in Private Streaming

Although there are a few deterministic streaming results (c.f. [12]), most streaming protocols employ the use of randomization. The amount of randomness required varies and typically ranges between pairwise and full independence. In particular, the streaming problem we consider in this paper requires $\mathcal{O}(n)$ fully independent random variables, where n is the stream size. Unfortunately, truly independence requires $\Omega(n)$ random bits, a prohibitive storage requirement for data streaming applications. In such cases, a common approach is

[‖] The gates result from the observation that given two independent coins with unknown probabilities $p, q \geq 1/\mathsf{poly}$, one can construct (in poly time) coins with probabilities $p \cdot q$, $1 - p$, or any convex combination of p, q.

to use a Pseudo-Random Generator (PRG) suitable for space-bounded computations. Indyk [7] pioneered this approach by using Nisan's PRG [15] construction, which fools space-bounded algorithms. An interesting property of the PRG is that it provides easy access to any bits of the pseudo-random pad. The property is used to ensure that any bit can be accessed efficiently every time it is requested; a critical part for streaming applications.

Unfortunately, space-fooling PRGs are not sufficient for functional privacy. In short, the security convention in Cryptography is to bound the adversary to $\mathsf{poly}(\kappa)$-time as opposed to $\mathcal{O}(\kappa)$-space for a security parameter κ. A typical adversary in the former model can break the randomness security in the latter (c.f. [3]).

In this paper, we consider a different approach. In a nutshell, we employ the use of a Pseudo-Random Function (PRF) [4] as follows. A brief review of PRF is informative. Let I_κ denote the set of all κ-bit strings. Consider H_κ the set of all functions from I_κ into I_κ (note that $|H_\kappa| = 2^{\kappa \cdot 2^\kappa}$). Let $F = \{F_\kappa\}$ be a function ensemble where F_κ assumes values from H_κ. Then, F is a PRF if it has the following properties: (a) indexing: each function in F_κ has a unique κ-bit index associated with it $F_\kappa = \{f_s | s \in I_\kappa\}$; (b) poly-time evaluation: $f_s(x)$ can be computed in $\mathsf{poly}(\kappa)$-time given $s \in I_\kappa$ and $x \in I_\kappa$; and (c) pseudo-randomness: no $\mathsf{poly}(\kappa)$-time probabilistic algorithm can distinguish the functions in F_κ from the ones in H_κ. Intuitively, given a κ-bit truly-random seed string s, a function f_s chosen from F_κ is as good as a random function to any $\mathsf{poly}(\kappa)$ adversary.

Many PRF constructions exist and suffice for our results. Our result in Section 5, however, uses the PRF construction of [13] because, to the best of our knowledge, it is currently the most efficient construction regarding the evaluation of $f_s(x)$.

4 Functional Privacy of Negligibly Biased Estimators

Consider a positive single-output deterministic function f with input size n. Our result is inspired in a technique implicit in the private protocol of [8]. We begin with a new definition.

DEFINITION 4.*[negligibly biased estimator (NBE)] A random variable X is a* negligibly biased *estimator for $f(x)$ in a parameter $\kappa \in \mathbb{N}$ if, for any admissible input x, $\mathsf{E}[X] \in (1 \pm \mu(\kappa))f(x)$ and* $\mathrm{Var}[X] < \infty$.

Observe that securely computing an NBE is not necessarily a functionally private approximation. Indeed, the higher moments of such computation depend on the input x. The following theorem attempts in squashing them and remove non-simulatable information.

THEOREM 5. *Suppose there exists an algorithm $\mathcal{A}(x, \epsilon', \delta')$ that $\langle 1/2, \mu(\kappa)\rangle$-approximates a positive function $f(x)$ with the following conditions. For any input x:*

a) $\mathcal{A}$ is a negligibly biased *estimator for $f(x)$ in a security parameter $\kappa \in \mathbb{N}$;*

b) $\exists$ an upper bound τ of $f(x)$, which is considered public knowledge.

Then, f admits a functionally private $\langle \epsilon, \delta\rangle$-approximation function such that:

1. it is computable in time $\mathcal{O}\big((\log\tau)(\kappa + \log(\log\tau) + \log(1/\delta)/\epsilon^2) \cdot T_{\mathcal{A}}(|x|, 1/2, \mu(\kappa))\big)$;

2. uses $\mathcal{O}\big((\log\tau + \log\kappa + \log\log[(1/2\delta)/\epsilon^2]) + S_{\mathcal{A}}(|x|, 1/2, \mu(\kappa))\big)$ of space,

where $T_{\mathcal{A}}(n, \epsilon', \delta')$ and $S_{\mathcal{A}}(n, \epsilon', \delta')$ are the running time and space usage of $\mathcal{A}(x, \epsilon, \delta)$ resp..

PROOF. We prove it constructively; i.e. we show how Function 1 achieves the claims. Let Bernoulli(q) represents a Bernoulli r.v. with success probability q.

```
Inputs:   input $x$ and parameters $\tau \geq f(x)$, $\epsilon \in (0,1)$, $\delta \in (0,1)$, security parameter $\kappa$,
          and access to a NBE $\mathcal{A}(x, \epsilon', \delta')$ for $\epsilon' = 1/2$ and $\delta' = \mu(\kappa)$.
Output:   a functionally private $\langle \epsilon, \delta \rangle$-approximation of $f(x)$
  1. Let $N = \Theta(\kappa + \log(\log \tau) + \log(2/\delta)/\epsilon^2)$
  2. For each iteration $i = 0, \ldots, \lceil \log \tau \rceil$:
      (a) Compute $Z_i = \sum_j^N Z_{i,j}$, where each $Z_{i,j}$ is the outcome of an independent trial of
              $\text{Bernoulli}\left(\frac{\mathcal{A}(x, 1/2, \mu(\kappa))}{(3/2)(\tau/2^i)}\right)$        (1)
          until iteration $\ell$ where $Z_\ell$ exceeds $N/8$.
      (b) Abort if any call to $\mathcal{A}(\cdot) > (3/2)(\tau/2^i)$ and output failure.
  3. Output $F = Z_\ell \cdot (3/2)(\tau/2^\ell)/N$
```

Function 1: Functionally private approximation function given an NBE.

Correctness. For each iteration $i = 0, 1, \ldots, \lceil \log \tau \rceil$, let the collection of r.v.s $\{X_{i,j}\}_{j \in [N]}$ represent the N independent outcomes of calling $\mathcal{A}(x, 1/2, \mu(\kappa))$. Each $X_{i,j}$ is an negligibly biased $\langle 1/2, \mu(\kappa) \rangle$-approximation of $f(x)$; i.e. with overwhelming probability in κ it holds that a sample from $X_{i,j} \in (1 \pm 1/2) f(x)$ and $\mathsf{E}[X_{i,j}] = (1 \pm \mu(\kappa)) f(x)$. As in Function 1, define Bernoulli r.v.s $\{Z_{i,j}\}_{j \in [N]}$ where each $Z_{i,j}$ has success probability $p_{i,j} = X_{i,j}/[(3/2)(\tau/2^i)]$.

Let $Z_i = \sum_j^N Z_{i,j}$. Also, let ℓ be the smallest index such that $Z_\ell > N/8$ as stated in Function 1 and let ℓ' be the index such that $\tau/2^{\ell'+1} \leq f(x) < \tau/2^{\ell'}$ (note that there is always such an index by definition of τ and iteration range of ℓ'). First, note that for any iteration $i = 0, 1, \ldots, \ell'$, $p_{i,j} \leq 1$ because $\tau/2^{\ell'} \geq f(x)$ and the confidence guarantees of $\mathcal{A}(\cdot)$ hold overwhelming in κ; i.e. only with $\mu(\kappa)$ probability, the protocol aborts and we can safely assume this does not happen. Therefore, all sample probabilities are proper in that range. We then show that $\ell \leq \ell'$ always holds; i.e. $Z_{\ell'} \geq N/8$ holds with overwhelming probability in κ. Indeed, the expectation of the Bernoulli trials at iteration ℓ' is

$$\mathsf{E}[Z_{\ell',j}] = \mathsf{E}\left[\frac{\mathcal{A}(x, 1/2, 2^{-\kappa})}{(3/2)(\tau/2^{\ell'})}\right] \geq \frac{\mathsf{E}[\mathcal{A}(x, 1/2, 2^{-\kappa})]}{(3/2)(2f(x))} = \frac{(1 \pm \mu(\kappa))f(x)}{3f(x)} \geq 1/4.$$

In turn, $\mathsf{E}[Z_{\ell'}] \geq N/4$ by linearity of expectations and thus

$$\mathsf{Pr}\left[Z_{\ell'} < N/8\right] \leq \mathsf{Pr}\left[Z_{\ell'} < (1/2)\mathsf{E}[Z_{\ell'}]\right] \leq \left(\frac{e^{-1/2}}{(1/2)^{(1/2)}}\right)^{\mathsf{E}[Z_{\ell'}]} \leq e^{-N/8} \leq \mu(\kappa),$$

which follows from a Chernoff bound and choice of N. Therefore, a suitable index $\ell \leq \ell'$ can be found in at most $\log(\tau) + 1$ iterations overwhelmingly in κ.

Now, recall that the output is $F = Z_\ell \cdot (3/2)(\tau/2^\ell)/N$. For the possible candidate exit iterations $i \leq \ell$, we have that

$$\mathsf{E}[Z_i] = \mathsf{E}\left[\sum_j^N Z_{i,j}\right] = \sum_j^N \mathsf{E}\left[\frac{\mathcal{A}(x, 1/2, 2^{-\kappa})}{(3/2)(\tau/2^i)}\right] = N\frac{f(x)}{(3/2)(\tau/2^i)} = \Theta(N).$$

Thus, by a Chernoff bound and union bound over the iterations,

$$\begin{aligned}\Pr\left[F > (1+\epsilon)f(x)\right] &= \log(\tau)\cdot\Pr\left[Z_i\cdot(3/2)(\tau/2^i)/N > (1+\epsilon)f(x)\right]\\ &= \log(\tau)\cdot\Pr\left[Z_i > (1+\epsilon)\mathsf{E}[Z_i]\right]\\ &\le \log(\tau)\cdot e^{-\Theta(N)\frac{\epsilon^2}{3}} \le e^{-(\kappa\epsilon^2+\log(2/\delta))} \le \delta/2.\end{aligned}$$

A similar result holds for $\Pr\left[F < (1-\epsilon)f(x)\right] \le \delta/2$. Therefore, we have shown that $\Pr\left[F \in (1\pm\epsilon)f(x)\right] \ge 1-\delta$ as desired. The running time follows from at most $\log(\tau)$ iterations of $\tilde{\mathcal{O}}(\kappa)$ independent samples of $T_{\mathcal{A}}(n, 1/2, \mu(\kappa))$. Space follows as one $\log\tau$-bit counter and one $\log N$-bit counter suffice for computing the Z_i's.

Privacy. F is functionally private to $f(x)$ as the Bernoulli trials can be simulated by an algorithm with similar skeleton as Function 1 but with success probabilities

$$p_{i,j} = \frac{f(x)}{(3/2)(\tau/2^i)}$$

instead in (1) (recall that $f(x)$ is given to the simulator, see Definition 3). Now, note that they are statistically indistinguishable from the protocol trials because each $X_{i,j} = \mathcal{A}(x, \epsilon', \delta')$ is a negligibly biased estimator of $f(x)$; i.e. $\mathsf{E}[\mathcal{A}(x,\epsilon',\delta')] = (1\pm\mu(\kappa))f(x)$ overwhelmingly in κ for $\epsilon' = 1/2$ and say $\delta' = 2^{-\Theta(\kappa)}$.** Indeed, the samples gathered until the last iteration ℓ were generated from proper probabilities (≤ 1) as argued earlier. Finally, recall that the higher moments of the Bernoulli random variables depend solely on its expectation —thus effectively squashing any non-simulatable higher moments of $\mathcal{A}(\cdot)$. Since τ is considered public, functional privacy is implied. ∎

Remark. The theorem is most useful when the upper bound τ is at most single-exponential in $f(x)$; as we shall see in the next section.

5 Functionally Private Streaming Approximation for the L_p Norm

The L_p norm, for $p \in (0, 2]$, of a vector $a \in \{-M, M\}^n$ is defined as $L_p(a) = ||a||_p = (\sum_i^n |a_i|^p)^{1/p}$. In this section, we prove the following theorem.

THEOREM 6. *There exists a* functionally private *$\langle\epsilon,\delta\rangle$-approximation of $||a||_p$, $p \in (0,2]$, in the streaming setting, requiring only $\mathcal{O}\left(\kappa^2\log^2(nM)(\kappa+\log(1/\delta)/\epsilon^2)\right)$ bits of space, and $\mathcal{O}\left(\kappa^2\log(nM)(\kappa+\log(1/\delta)/\epsilon^2)\right)$ update and $\mathcal{O}\left(\kappa\log^2(nM)(\kappa+\log(1/\delta)/\epsilon^2)\right)$ update query time for arbitrary $\epsilon, \delta \in (0,1)$ and security parameter κ.*

Before proceeding, it is instructive to recall the estimator of [11].

Geometric Mean Unbiased Estimator for L_p [11]. Let $\mathbf{R}$ be the $\mathbb{R}^{\ell\times n}$ projection matrix with i.i.d. entries $R_{i,j} \sim S(p,1)$, where $S(p,\gamma)$ denotes a discretized symmetric p-stable distribution over $\mathbb{R}$ with scale parameter γ. Let $x = \mathbf{R}a$ be the "sketch" of a as $\ell \ll n$ (ℓ is set later).

** let us not confuse ϵ' and δ' with ϵ and δ. The former parameters are the ones used for invoking the NBE $\mathcal{A}$, while the latter are the error and confidence parameters of the functionally private approximation function.

By the properties of the distribution, each $x_j = \sum_i a_i R_{i,j} \sim ||a||_p X_j$, where $X_j \sim S(p,1)$. Equivalently, we can write $x_j \sim S(p, ||a||_p)$. Such distributions exists for $p \in (0,2]$. Thus, to estimate $||a||_p$, it boils down to approximating the scale parameter γ from ℓ i.i.d. samples. In [7], the author proposed using the estimator median$(|x_1|, |x_2|, \ldots, |x_\ell|)$. However, [11] has shown that not only it is severely biased but also hard to bias-correct it analytically or algorithmically. Therefore, for $p \in (0,2]$, [11] proposed using a bias-corrected version of the *geometric mean* estimator:

$$\hat{L}_{p,\mathrm{gm}} = \prod_{j=1}^{\ell} |x_j|^{1/\ell} \Big/ \left[\frac{2}{\pi}\Gamma\left(\frac{p}{\ell}\right)\Gamma\left(1-\frac{1}{\ell}\right)\sin\left(\frac{\pi}{2}\frac{p}{\ell}\right)\right]^{\ell}, \tag{2}$$

where $\Gamma(z)$ is the Gamma function of a real-valued z. The estimator is strictly unbiased, or $\mathsf{E}[\hat{L}_{p,\mathrm{gm}}] = ||a||_p$. Moreover, it has finite variance and exponential tail bounds, crucial for an $\langle \epsilon, \delta \rangle$-approximation of $||a||_p$ for arbitrary $\epsilon, \delta \in (0,1)$.

The correctness of the construction relies on building the projection matrix $\mathbf{R}$ from truly random samples. Unfortunately, that requires $\Omega(n\ell)$ bits of storage. By using the Pseudo-Random Function construction of [13] instead (see Section 3.1) we only need to store a κ-bit seed s_j per each sample $j \in [\ell]$. This is correct as long as $\kappa = \Omega(\log n)$ because we use the vector coordinate $i \in \{0,1\}^{\log n}$ as input for the PRF given seed s_j.

Proof of Theorem 6. We transform the *unbiased geometric estimator* $\hat{L}_{p,\mathrm{gm}}$ of (2) to an NBE with $\Omega(\kappa)$-bit precision. The theorem then follows by applying Theorem 5.

Specifically, for $p = 1$, the denominator in (2) simplifies to $[2\sin(\pi/2\ell)/\sin(\pi/\ell)]^{\ell} = 1/\cos^{\ell}(\pi/2\ell)$. It is known that it suffices to use $\mathcal{O}(\log 1/\epsilon)$ terms to $(1 \pm \epsilon)$-approximate $\cos^{\ell}(x)$ (by bounding the Taylor polynomial), or in our case $\mathcal{O}(\kappa\ell)$ terms to $(1 \pm \mu(\kappa))$-approximate For $p = 2$, the same denominator simplifies to $[p\Gamma(1/\ell)/\Gamma(1/p\ell)]^{\ell}$. Approximating it negligibly in κ implies getting an $(1 \pm 2^{-\kappa\ell})$ approximation to the Gamma function (note the power ℓ). A result from [17] does so with $\mathcal{O}(\kappa\ell)$ time with relative error $2^{-\kappa\ell}$. A similar argument applies for $p \in (0,2]$. Finally, observe that for agreed-upon values of κ, ϵ, and δ, the correction factor can be pre-computed (the theorem claims assume this fact).

Now, we validate our storage claims. Recall that Theorem 5 makes at most $\mathcal{O}(\log \tau) \cdot N$ invocations to the NBE $\mathcal{A}$, where τ is an upper bound on $f(x)$ and $N = \Theta(\kappa + \log(\log \tau) + \log(1/\delta)/\epsilon^2)$. Since $\tau \leq nM^2$ (for any $p \in (0,2]$)) we have $\mathcal{O}\big(\log(nM)(\kappa + \log(1/\delta)/\epsilon^2)\big)$ invocations of $\mathcal{A}$.[††] On the other hand, each invocation of $\mathcal{A}$ requires taking ℓ samples (or sketches). In [11], it was shown that setting $\ell = \mathcal{O}\big(\log(1/\delta)/\epsilon^2\big)$ suffices for an $\langle \epsilon, \delta \rangle$-approximation of $||a||_p$ using (2). Since $\mathcal{A}$ is called with $\epsilon' = 1/2$ and $\delta' = \mu(\kappa) = 2^{-\Theta(\kappa)}$ in Theorem 5, we have that each invocation requires $\ell = \mathcal{O}\Big(\log(1/2^{-\Theta(\kappa)})/(1/2)^2\Big) = \mathcal{O}(\kappa)$ sketches. Therefore, multiplying the number of invocations by the number of samples we get that the total storage requirement is $\mathcal{O}\big(\kappa \log(nM)(\kappa + \log(1/\delta)/\epsilon^2)\big)$ sketches. Each of these sketches require a counter and a κ-bit seed for the PRF. The former requires $\mathcal{O}(\log(nM))$ bits as the maximum value for $f(x)$ is nM^2 for any $p \in (0,2]$. Thus, the total storage is $\mathcal{O}\Big(\kappa^2 \log^2(nM)(\kappa + \log(1/\delta)/\epsilon^2)\Big)$ bits as desired.

[††] assuming $\kappa = \Omega(\log\log\tau) = \Omega(\log(\log nM))$.

The amount of computation per update per sketch is dominated by κ modulo multiplications and one exponentiation of κ-bit numbers when using the PRF construction of [13]. These can be performed in $\mathcal{O}(\kappa)$ constant-time computations. The update time is thus simply $\mathcal{O}\big(\kappa^2 \log(nM)(\kappa + \log(1/\delta)/\epsilon^2)\big)$ as desired assuming that operations on $\mathcal{O}(\log(nM))$-bit strings are constant. Finally, the query time is $\mathcal{O}\big(\kappa^2 \log(nM)(\kappa + \log(1/\delta)/\epsilon^2)\big)$ as the work of Function 1 is simply linear in the storage size once all sketches are available. ■

6 Private Approximation of $\sharp$P-complete problems

Consider the following abstract problem. Let U be a finite set whose elements are binary strings of size n. Let the Boolean function $h : U \to \{0,1\}$ partition U. The goal is to estimate the cardinality of $D = \{u | u \in U \wedge h(u) = 1\}$. Most problems in $\sharp$P can be formulated as the problem above. Indeed, $\sharp$P can be seen as the class of function problems counting the number of accepting paths in an NP machine [18]. In this section, we focus on obtaining efficient (read poly-time) functionally private approximations to the above abstract problem as exact solutions are typically not feasible.

Monte-Carlo sampling methods are useful in estimating $\mu = |D|/|U|$. From Chernoff bounds, an $\langle \epsilon, \delta \rangle$-approximation is possible using $\tilde{\mathcal{O}}(1/\mu)$ independent samples of $h(u)$ for an u chosen uniformly at random (u.a.r.) from U. An *efficient* algorithm, however, requires that $\mu \geq 1/\mathsf{poly}$ provided that it is poly-time computable to sample an element u u.a.r. from U and compute $h(u)$. Unfortunately, μ may be exponentially small in n, requiring a prohibitive super-polynomial samples. An alternative approach is the method of Karp and Luby [9]. The crux is on finding a small enough multiset V, containing all elements of D, such that $\mu = |D|/|V|$ is large enough for efficient sampling. The following theorem summarizes their *coverage algorithm*, as it is known, for an abstract Union of Sets problem.

THEOREM 7.*[Karp and Luby [9]] Let U and D be defined as before. Suppose there are sets $\{D_1, \ldots, D_m\} \subseteq D$ s.t. $D = \bigcup_i^m D_i$ and the following conditions hold, $\forall i \in [m]$:*

1. *$|D_i|$ can be computed in $\mathsf{poly}(n,m)$ time;*
2. *any element $s \in D_i$ can be sampled u.a.r. from D_i in $\mathsf{poly}(n,m)$ time;*
3. *given any $s \in D$, it can be decided if $s \in D_i$ in $\mathsf{poly}(n,m)$ time.*

Then, an $\langle \epsilon, \delta \rangle$-approximation for $|D|$ can be computed in $\mathsf{poly}(n, m, 1/\epsilon, \log(1/\delta))$ time.

Private Coverage Algorithm. What prevents the *coverage algorithm* from being functionally private to f using current techniques is the fact that $|V|$ depends on x. Indeed, $|V|$ cannot be inferred from $f(x)$ alone and thus the higher moments of the distribution induced by X depends on the structure of x and thus breaks functional privacy (c.f. Section 2).

Let X_j be a Bernoulli r.v. representing the j^{th} sample of a coin with success probability $p = |D|/|V|$ as in the proof of Theorem 7 [9]. Alternatively, one might be tempted to construct an event "$Y_j = 1$" where Y_j is a Bernoulli r.v. with probability $q = |V|/\tau$ and sample from the joint Bernoulli distribution $\mathsf{E}\big[\text{"}X_j = 1\text{" and "}Y_j = 1\text{"}\big] = p \cdot q = \frac{|D|}{|V|} \cdot \frac{|V|}{\tau} = \frac{|D|}{\tau}$ for a publicly known value τ (or one that can be inferred from $f(x)$), where $p = \mathsf{E}\big[X_j\big]$. That way the output distribution depends solely on $|D|$ (and no-harm τ) and functional privacy is implied by the feasibility results of [1] using their formula $f(x) = \psi(\phi(\cdot))$ where

$\phi = p \cdot q$ and $\psi(n) = n \cdot \tau$ (see Theorem 6.4 in [1]). However, we note that in this case τ must be larger than $|V|$ so that the coin is proper. Unfortunately, the only known upper bound on $|V|$ we know of without knowing x is $m2^n$ as every element can be part of each set D_i. In such case, $q = |V|/(m2^n) < 1/\mathsf{poly}$ for small values of $|V|$ and no efficient sampling is possible.

Our approach instead is to squash the higher moments of X to prevent non-simulatable information from leaking. To that end, we use the unbiased coverage algorithm of Theorem 7 as the *negligibly biased* estimator in our main theorem, Theorem 5. The result is below.

THEOREM 8. *Let U, D and V and the set forth conditions on them be as in Theorem 7. Furthermore, suppose there exists a publicly known upper bound τ on $f(x)$. Then there exists a functionally private $\langle \epsilon, \delta \rangle$-approximation for $|D|$ in $\mathsf{poly}(\kappa, n, m, \log \tau, 1/\epsilon, \log(1/\delta))$ time for a security parameter κ.*

PROOF. Let $\mathcal{A}(x, \epsilon, \delta)$ be the coverage algorithm of Theorem 7. The theorem follows from a direct application of Theorem 5 using $\mathcal{A}$ with parameters $\epsilon = 1/2$ and $\delta = \mu(\kappa)$ and upper bound $\tau = 2^n$. ∎

Private $\sharp$DNF. Let $F = \bigvee_i^m C_i$, be a propositional formula in *disjunctive normal form* where each C_i is a conjunction of a subset of literals defined with respect to n Boolean variables $x_1, \ldots, x_n$. The goal is to output the number of satisfiable assignments to F, or $\sharp F$. The problems is $\sharp$P-complete [18]. In [9], Karp and Luby also showed a connection between the abstract Union of Sets problem and $\sharp$DNF. Our result below uses this connection.

COROLLARY 9. *There exists a functionally private $\langle \epsilon, \delta \rangle$-approximation for $\sharp$DNF computable in $\mathsf{poly}(n, m, 1/\epsilon, \log(1/\delta))$ time.*

PROOF. The claims follows directly from Theorem 8. Essentially, we show set $D = \bigcup_i^m D_i$ can be built as required and the conditions put forth in Theorem 7 (and Theorem 8) hold. Let each D_i be the set of assignments satisfying clause C_i. Then, clearly $\sharp F = |D|$. The conditions are met as follows, $\forall i \in [m]$: 1) $|D_i|$ can be computed in $\mathcal{O}(1)$ as $|D_i| = 2^{n-|C_i|}$; 2) sampling an element $s \in D_i$ u.a.r. from D_i requires setting the proper assignments for the literals in C_i and choosing u.a.r. from $\{\mathsf{true}, \mathsf{false}\}$ for the other literals not in C_i; and 3) trivial to evaluate whether or not $s \in D_i$ for any $s \in D$ in $\mathcal{O}(n)$ time. The corollary follows. ∎

Further Applications. In [16], it was shown that $\sharp k \log$DNF (a special case of $\sharp$DNF restricting the formula to at most $k \log n$ variables per disjunct.) and $\sharp$DNF are complete for classes $\sharp\Sigma_1$ and $\sharp\mathsf{R}\Sigma_2$ respectively. These are logic-based classes of counting problems. The problems are complete under a product reduction, which is a reduction from f to g where $\exists \phi, h \in \mathsf{FP}$, $h : \mathbb{N} \to \mathbb{N}$ such that $\forall x, f(x) = g(\phi(x)) \cdot h(|x|)$, with FP being the complexity class of polynomial-time computable functions problems.

Observe that the reduction is private and approximation-preserving. Note that $h(|x|)$ not only preserves approximability but also does not leak anything about x. We conclude that a functionally private $\langle \epsilon, \delta \rangle$-approximation to g implies one to f. Consequently, we have that all problems in $\sharp\Sigma_1$ and $\sharp\mathsf{R}\Sigma_2$ can be privately approximated, including problems such as $\sharp$NON-VERTEX-COVERS, $\sharp$NON-CLIQUES, $\sharp$NON-DOMINATING-SETS, and $\sharp$NON-HITTING-SETS to cite a few (c.f. [16]). We defer details to the full version.

References

[1] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure Multiparty Computation of Approximations. *ACM Transactions on Algorithms (TALG)*, 2(3):435–472, 2006.

[2] Michael Freedman, Kobbi Nissim, and Benny Pinkas. Efficient Private Matching and Set Intersection. In *EUROCRYPT'04: Advances in Cryptology*, pages 1–19, 2004.

[3] Oded Goldreich. *Foundations of Cryptography: Volume II, Basic Applications*, volume 2. Cambridge University Press, New York, NY, USA, July 2004.

[4] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, 1986.

[5] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In *STOC'87*, pages 218–229, New York, NY, USA, 1987. ACM.

[6] Shai Halevi, Robert Krauthgamer, Eyal Kushilevitz, and Kobbi Nissim. Private Approximation of NP-hard Functions. In *STOC'01*, pages 550–559, 2001.

[7] Piotr Indyk. Stable Distributions, Pseudorandom Generators, Embeddings, and Data Stream Computation. *Journal of the ACM*, 53(3):307–323, 2006.

[8] Piotr Indyk and David P. Woodruff. Polylogarithmic private approximations and efficient matching. In *TCC'06*, volume 3876, pages 245–264, 2006.

[9] Richard M. Karp and Michael Luby. Monte-Carlo Algorithms for Enumeration and Reliability Problems. In *SFCS'83*, pages 56–64. IEEE Computer Society, 1983. An extended abstract appeared in FOCS'97.

[10] Joe Kilian, André Madeira, Martin J. Strauss, and Xuan Zheng. Fast Private Norm Estimation and Heavy Hitters. In *TCC'08: Proceedings of the Fifth Theory of Cryptography Conference*, pages 176–193, New York, NY, USA, 2008. Springer Berlin/Heidelberg.

[11] Ping Li. Estimators and Tail Bounds for Dimension Reduction in $l_\alpha(0 < \alpha \leq 2)$ Using Stable Random Projections. In *SODA'08*, pages 10–19, Philadelphia, PA, USA, 2008.

[12] S. Muthukrishnan. *Data Streams: Algorithms and Applications*, volume 1. Now Publishers, August 2005.

[13] Moni Naor and Omer Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, 51(2):231–262, 2004.

[14] J. Nelson and D. P. Woodruff. Revisiting Norm Estimation in Data Streams. *ArXiv e-prints*, nov 2008.

[15] N. Nisan. Pseudorandom Generators for Space-Bounded Computations. In *STOC'90*, pages 204–212. ACM, 1990.

[16] Sanjeev Saluja, K. V. Subrahmanyam, and Madhukar N. Thakur. Descriptive Complexity of ♯P Functions. *Journal of Computer and System Sciences*, 50(3):493–505, June 1995.

[17] J. L. Spouge. Computation of the gamma, digamma, and trigamma functions. *SIAM Journal on Numerical Analysis*, pages 931–944, 1994.

[18] Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[19] Andrew C. Yao. Protocols for Secure Computations. In *SFCS'82*, pages 160–164, 1982.