Secure Multiparty Computation of Approximations

Joan Feigenbaum *
Yuval Ishai **
Tal Malkin ***
Kobbi Nissim †
Martin Strauss ‡
Rebecca N. Wright §

Abstract. Approximation algorithms can sometimes be used to obtain efficient solutions where no efficient exact computation is known. In particular, approximations are often useful in a distributed setting where the inputs are held by different parties and are extremely large. Furthermore, for some applications, the parties want to cooperate to compute a function of their inputs without revealing more information than they have to.

Suppose the function $\hat{f}$ is an approximation to the function f. Secure multiparty computation of f allows the parties to compute f without revealing more than they have to, but it requires some additional overhead in computation and communication. Hence, if computation of f is inefficient or just efficient enough to be practical, then secure computation of f may be impractically expensive. Furthermore, a secure computation of $\hat{f}$ is not necessarily as private as a secure computation of f, because the output of $\hat{f}$ may reveal more information than the output of f. In this paper, we present definitions and protocols of secure multiparty approximate computation that show how to realize most of the cost savings available by using $\hat{f}$ instead of f without losing the privacy of a secure computation of f.

We make three contributions. First, we give formal definitions of secure multiparty approximate computations. Second, we present an efficient, sublinear-communication, private approximate computation for the Hamming distance; we also give an efficient, polylogarithmic-communication solution for the L^2 distance in a relaxed model. Finally, we give an efficient private approximation of the permanent and other related #P-hard problems.

Keywords: secure multiparty computation, cryptographic protocols, approximation algorithms, massive data sets, streaming algorithms, Hamming distance.

* Yale University, Computer Science Department, PO Box 208285, New Haven, CT 06520-8285. Part of this work was done while the author was at AT&T Labs—Research. joan.feigenbaum@yale.edu
** DIMACS and AT&T Labs—Research. yuval@dimacs.rutgers.edu
*** AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 USA. tal@research.att.com
† Weizmann Institute of Science, Dept. of Computer Science and Applied Math. Rehovot, Israel. Part of this work was done while the author was at AT&T Labs—Research. kobbi@wisdom.weizmann.ac.il
‡ AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 USA. mstrauss@research.att.com
§ AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 USA. rwright@research.att.com

1 Introduction

There is an increasing number and variety of real-world applications that collect a massive amount of data and wish to make use of that data. For example, massive data sets arise in physical sciences such as biology and astronomy, in marketing, in network operations, and in Web searches. The search for efficient and effective data mining algorithms, particularly of massive data sets, is an important emerging area of research (cf. [DIM97,DIM00] and the many activities described therein).

Unfortunately, many useful functions are expensive to compute. Even functions that are efficiently computable for moderately sized data sets are often not efficiently computable for massive data sets. For example, even quadratic algorithms cannot generally be considered practical on input consisting of a terabyte of data; such data sets are now routinely generated daily.[1] Besides the size of the input, the distributed nature of data sets can be an obstacle to computation. A single logical data set is often stored in several remote pieces, and a computation on the data set may require prohibitively costly communication among the different storage sites. For example, several elements of a large wide-area network may gather statistics independently; these statistics may need to be sent to a central location to be analyzed.

Another important concern, in addition to the efficiency of a computation, is its security. In a distributed setting, the pieces of a distributed data set may be controlled by different parties who wish to collaborate in order to compute some function of their data without fully revealing their piece of the data to the other parties. To that end, the parties may want to compute a function of their inputs securely, i.e., so that no party learns anything about the others' inputs except what is implied by her own output, perhaps even if some of the other parties behave maliciously. For example, rival Internet service providers often strike "peering agreements," in which each carries the other's Internet traffic at no cost, as long as the characteristics of the traffic carried by each peer for the other are comparable. The prospective peers each have data sets describing the characteristics of their own traffic, and they would like to verify the similarity of these data sets without revealing more than they have to. Several recent papers have considered the problem of privacy-preserving data mining [AR00,LP00], recognizing that it is often desirable to perform data mining without revealing unnecessary information about the data.

Each of the above two concerns has been previously addressed (separately): On one hand, when the cost of an exact computation of a function f is too high, the parties may use an *approximation* $\hat{f}$ to f. In some cases, the communication of only a small random sample from each part of a data set stored in remote pieces suffices for an approximation. In other cases, the communication of the result of a local computation depending on the entire local data set is sufficient. In both situations, the computation on the samples typically requires less communication and less computation than an exact computation on the original data set. On the other hand, *secure multiparty computation* (initiated by [Yao82,GMW87,BGW88,CCD88]) allows a group of parties to compute a function f in such a way that no more is revealed to a party about other parties' inputs and outputs than what is implied by her own input and output.

In this paper we address both concerns simultaneously: our goal is to construct approximation algorithms (more efficient than exact computation), which also maintain the privacy of the data. Note that the straightforward approach of simply computing an approximation $\hat{f}$ via a secure multiparty computation, will *not* work, as it will not necessarily be private, nor efficient. Indeed, even when $\hat{f}$ itself is efficient, computing it using a general-purpose secure computation is typically very expensive. Moreover, a secure computation of $\hat{f}$ may still leak information about f, that follows from the output of $\hat{f}$ itself. To illustrate this, consider a function f and an approximation $\hat{f}$ to f that outputs $f(x_1, \ldots, x_n)$ with the last bit possibly flipped so that that last bit is 0 if x_1 is even and 1 if x_1 is odd. Then $\hat{f}(x_1, \ldots, x_n)$ is a good approximation but unnecessarily reveals the parity of x_1.

Our work

In this paper, we provide definitions of secure approximate multiparty computation that disallow the problems of information leakage discussed above, and we present protocols for several natural functions.

For massive data sets, *distance* functions are particularly important because they give a measure of similarity between two data sets. For example, telephone companies may want to compute joint statistics on their

[1] For example, AT&T's phone network carries about 300 million calls per day, each of which generates a few kilobytes of billing data.

calling data, ISPs may want to verify similar "peering" traffic characteristics, and Web search companies may want to compare their images of the Web. Because the exact computation of the Hamming and related distances requires $\Omega(n)$ communication, there has been much recent work on sublinear communication distance approximations. For example, Alon et al. [AMS96] and Feigenbaum et al. [FKSV99] present algorithms for efficiently approximating the L^2 and L^1 distances, respectively, between two massive data sets. However, these L^p approximations also suffer the kind of information leakage described above. In this paper, our main technical contribution is an efficient Hamming distance approximation that does not leak any unnecessary information about the parties inputs. If the model is relaxed to allow linear offline computation (before the parties know their inputs), we also give a private approximation for the L^2 norm (the Hamming distance is a special case), whose online communication is extremely efficient (polylogarithmic).

Approximation algorithms are also useful in the setting where the data involved is only moderate in size, but the function itself is computationally difficult. We also consider this case, and we provide a private approximation to natural #P-hard problems, related to the permanent.

To summarize, the main contributions of this paper are as follows:

- definitions of secure and private approximate multiparty computation;
- a sublinear-communication private approximation for the Hamming distance between two bit strings;
- a private approximation to natural #P-hard problems, related to the permanent.

We formalize the definition of secure multiparty approximation in Section 2. In Section 3, we present efficient private approximations for the Hamming distance. In Section 4, we present efficient private approximations of #P-hard problems. We conclude in Section 5. Due to space considerations, we defer several technical discussions to an appendix. In Appendix A, we give some motivation for our definition and discuss other candidate definitions. In Appendix B, we give some alternate protocols for our results, including our private approximation of the L^2 norm in the alternate off-line communication model. In Appendix C, we give proofs of lemmas and theorems not given in the main body.

Related results

Several existing approximation algorithms (cf. [AGMS99,FKSV99,FS00,KN97,Ind00]) for the L^p or Hamming distance are efficient even for massive data sets. These algorithms all use correlated randomness between players to reduce the communication required (more details are given in Section 3.1). However, these results do not directly translate into communication-efficient private approximation protocols for a variety of reasons, as is discussed further in Section 3.2.

Since the initial presentation of an early version of this work [FFSW00], there has already been further interest and results. Halevi et al. [HKKN01] investigate private approximations of NP-hard functions. They show that there exist natural NP-hard functions (such as the size of the minimum vertex cover in a graph) which do not admit non-trivial private approximation, although they admit good approximation algorithms without the privacy restriction. They further show that this phenomenon does not hold for all NP-hard functions, by presenting an artificial NP-hard function that can be privately approximated. In our paper (Section 4), we give examples of natural #P-hard functions that admit private approximation.

The approach of constructing private sublinear-communication protocols, avoiding the transformation from a circuit [Yao82], was initiated in the context of private information retrieval [CGKS95] and further studied both in other specific contexts (e.g., [LP00]) and in more general contexts [NN01]. The latter work presents a generic methodology for transforming protocols in the communication complexity model into private protocols with low communication overhead. However, a straightforward application of their techniques to existing Hamming distance approximation protocols results in a protocol requiring superpolynomial work.

2 Secure Multiparty Approximations

In this section we define the notion of secure multiparty approximation. We first address each of the two components separately: *approximation* and *secure multiparty computation*, giving background and notation. We then present our definition of *secure approximations*. Some further discussion of other candidate definitions (and, in some cases, their pitfalls) may be found in Appendix A.

2.1 Approximations

An *approximation requirement* is any binary relation $\mathcal{P}$ between a deterministic real-valued function f, called the *target function*, and a possibly randomized real-valued function $\hat{f}$, called the *approximating function*. It defines which functions are considered good approximations. We say that $\hat{f}$ is a *$\mathcal{P}$-approximation to f* if $\mathcal{P}(f, \hat{f})$ holds. We say that an algorithm or a protocol *$\mathcal{P}$-approximates f* if it outputs some $\mathcal{P}$-approximation of f. The most common approximation relation, referred to as $\langle \epsilon, \delta \rangle$-approximation, is defined as follows.

Definition 1. *We say that $\hat{f}$ is an* $\langle \epsilon, \delta \rangle$-approximation *of f if for all inputs* $\mathbf{x}$*,*

$$\Pr[(1-\epsilon)f(\mathbf{x}) \leq \hat{f}(\mathbf{x}) \leq (1+\epsilon)f(\mathbf{x})] \geq 1-\delta,$$

where the probability is over the randomness of $\hat{f}$.

2.2 Secure Multiparty Computation

Secure multiparty computation allows two or more parties to evaluate some function of their inputs, such that no more is revealed to a party or a set of parties about other parties' inputs and outputs, except what is implied by their own inputs and outputs. When formally defining security, it is convenient to think of an *adversary* which tries to gain as much advantage as it can by corrupting at most t parties during the execution of the protocol. Security is then defined by requiring that whatever the adversary achieves in a *real-life* execution of the protocol it can efficiently *simulate* while corrupting at most t parties in an *ideal model*, in which a trusted party is being used to evaluate the function. Thus, the protocol prevents the adversary from gaining an extra advantage over what it could have gained in an ideal solution utilizing a trusted party.

There are several notions of security with various degrees of strength. We refer the reader to, e.g., [Gol98,Can00,Bea91,MR91] for formal definitions of secure computation in different settings. In this work we will mostly deal with the special case of *private computation*, which assumes that the adversary is *passive* and cannot modify the behavior of corrupted parties.[2] In particular, private computation is only concerned with the information learned by the adversary, and not with the effect it may have on the protocol's correctness. Our general definitions, however, will apply also to the case of an *active* adversary. We will use the term "secure" rather than "private" whenever the discussion applies to both the active and the passive case.

Another distinction between different notions of security is the *emulation quality*, which determines the extent to which the output produced by the simulator in the ideal model should resemble the view of the adversary in the real-life execution of the protocol. The three standard variants of emulation quality considered in the literature are *perfect*, *statistical*, and *computational* indistinguishability. These naturally define corresponding notions of perfect, statistical, and computational security. In this work we will mostly be concerned with the two-party case, in which only computational security can be achieved. However, our definitions apply to the other variants as well.

In the following we review a somewhat simplified definition of a private 2-party protocol. We refer the reader to, e.g., [Gol98,Can00], for a more thorough and general treatment.

The two parties will be referred to as Alice and Bob. The functionality of the protocol is specified by a (possibly randomized) mapping g from a pair of inputs (a, b) to a pair of outputs (c, d). A randomized, synchronous protocol π proceeds by rounds, where in each round each party may send to the other party a message based on a security parameter k, its input, its random input, and messages received in previous rounds. At each round, each party may decide to terminate and output some value based on its entire view. Both parties are assumed to be polynomial in the security parameter k.[3]

For defining the privacy of π with respect to g, it is convenient to use the following notation. Let $\text{REAL}_{\pi,A}(k, \mathbf{x})$ be a random variable containing the view of Alice in π on input $\mathbf{x} = (a, b)$ with security

[2] A passive adversary can also model *honest-but-curious* parties, which follow the protocol but may try to infer additional information from the messages they see.

[3] This requirement effectively implies that the length of the input must be polynomial in the security parameter. This standard requirement has to be relaxed in order to capture protocols where the security parameter can be smaller than any polynomial (e.g., poly-logarithmic) in the total length of the input. We refer the reader to [CMS99] for such definitions.

parameter k, *concatenated* with the output of Bob. Alice's view includes her inputs, random inputs, and all messages exchanged. (The concatenation of this view with the output of Bob serves to capture the requirement that Alice should not learn additional information on the output of Bob; it is redundant in the case that g is determinstic.) A similar notation is defined for Bob. For any efficient algorithm $\mathcal{S}$, denote by $\text{IDEAL}_{\pi,A,\mathcal{S},g}(k,(a,b))$ the output of the following process: (1) apply g to $\mathbf{x}$, resulting in a pair of outputs (c,d); (2) invoke $\mathcal{S}$ on (k,a,c); (3) concatenate the output of $\mathcal{S}$ with d. (The above process models the interaction of an adversary corrupting Alice with the ideal model for evaluating g.) Again, this notation is defined symmetrically for Bob.

Definition 2. *A protocol π is a* private protocol computing g *if the following properties hold:*

Correctness. *The pair of outputs are computationally indistinguishable from $g(a,b)$.*

Privacy. *The view of an adversary corrupting a party in the real-life protocol can be simulated in an ideal model involving a trusted party. We formalize the simulation requirement for Alice; the second simulation requirement for Bob is symmetrical. We require the existence of an efficient simulator $\mathcal{S}$, such that for any inputs $\mathbf{x}=(a,b)$, the distribution ensembles $\{\text{REAL}_{\pi,A}(k,\mathbf{x})\}_{k\in\mathbb{N}}$ and $\{\text{IDEAL}_{\pi,A,\mathcal{S},g}(k,\mathbf{x})\}_{k\in\mathbb{N}}$ are computationally indistinguishable. That is, for any family $\{C_k\}$ of polynomial-size circuits, any constant $c>0$, any sufficiently large k and and any $\mathbf{x}$,*

$$|\Pr[C_k(\text{REAL}_{\pi,A}(k,\mathbf{x}))=1]-\Pr[C_k(\text{IDEAL}_{\pi,A,\mathcal{S},g}(k,\mathbf{x}))]=1|<k^{-c}$$

It is known that any polynomial-time computable (randomized) functionality can be efficiently and privately computed [Yao82,GMW87]. This general feasibility result can be based on the existence of trapdoor permutations, which will be assumed throughout the paper.

2.3 Secure Approximations

We turn to the question of defining secure approximations. To preclude the computation of an approximation from leaking unnecessary information, our definitions require not only that the computation of the approximate output does not reveal more about other parties' inputs and outputs than that approximate output, but also that the approximate output itself does not reveal more about other parties' inputs and outputs than the exact output does. We restrict our attention to an approximation of a *deterministic* function f, mapping an input $\mathbf{x}=(x_1,\ldots,x_m)$ (where x_i is the i-th party's input) to a non-negative integer y.[4] A generalization to multi-output functions is straightforward.

We will start by defining a notion of *functional privacy* on which our definition will rely. Informally, we say that a (possibly randomized) approximation function $\hat{f}$ is functionally private with respect to the target function f, if the output of $\hat{f}$ reveals no more information on its input than f does. Note that this is an inherent property of the function $\hat{f}$ (rather than of a particular protocol that computes $\hat{f}$). This is formalized as follows:

Definition 3. *Let $f(\mathbf{x})$ be as above, and let $\hat{f}(\mathbf{x})$ be a possibly randomized function. We say that $\hat{f}$ is* functionally t-private with respect to f *if there exists an efficient randomized algorithm $\mathcal{S}$, called a* simulator*, such that for every input $\mathbf{x}$ and $1\le i_1,\ldots,i_t\le m$, $\mathcal{S}((i_1,x_{i_1}),\ldots,(i_t,x_{i_t}),f(\mathbf{x}))$ is identically distributed to $\hat{f}(\mathbf{x})$.*

Thus $\hat{f}$ is functionally 0-private if it has a simulator S such that $S(f(\mathbf{x}))$ and $\hat{f}(\mathbf{x})$ are identically distributed. In that case, we call $\hat{f}$ *functionally private* with respect to f. Note that functional privacy implies functional t-privacy, for any $t\ge 0$. Our examples in this paper are all functionally private.

Our definition for secure approximation requires that the protocol securely computes some functionally-private approximation $\hat{f}$ of f. Since we defined $\hat{f}$ to be a single-output function, we will need to fix some convention for extending it to a multi-output function. Our default interpretation of a single-output function $\hat{f}$ in a multi-party setting assumes that a single value y is sampled from $\hat{f}(\mathbf{x})$ and is output by all parties. We stress that other conventions are possible (see Remark 1), and a more general treatment would allow specifying an admissible collection of multi-output approximations. However, we prefer here simplicity over generality. The above discussion is formalized by the following definition, which may be instantiated with any notion of security (e.g., active or passive adversary, and computational, statistical, or perfect emulation).

[4] We will also be concerned with approximations of real-valued functions. In this case, it is implicitly assumed that the outputs of f admit some succinct representation.

Definition 4. *Let f be a deterministic function. The protocol π is a* t-secure $\mathcal{P}$-approximation protocol for *f if it securely computes a (possibly randomized) function $\hat{f}$, such that $\hat{f}$ is both functionally t-private with respect to f and a $\mathcal{P}$-approximation of f.*

Intuitively, the functional privacy of $\hat{f}$ with respect to f says that the input/output relation of the protocol does not reveal anything except what would have been revealed by learning f, while the secure computation of $\hat{f}$ ensures that nothing additional is revealed during the computation.

Remark 1. As noted above, we assume by default that $\hat{f}$ induces an m-output functionality in which each party receives an *identical* instance of $\hat{f}(\mathbf{x})$. This convention has the advantage of guaranteeing consistency between the outputs. However, from the privacy point of view, it may be desirable to insist that outputs of different parties be *independently* distributed according to $\hat{f}(\mathbf{x})$. This prevents the parties from learning additional information about the *outputs* of other parties, except what would follow by learning $f(\mathbf{x})$. Similarly, some proper subset may learn identical or independent samples of $\hat{f}(\mathbf{x})$, while the others learn nothing. We do not impose any specific preference among the options, and note that our protocols can easily be modified to be secure under the appropriately-modified definition as well.

Approximations are useful both for settings in which the inputs are small but the target function is intractable and for settings in which the inputs are massive. In the setting of massive inputs, we will consider sublinear approximations to functions whose exact computation requires at least linear and at most polynomial resources. We consider several resources, all of which have been considered previously, for example by [AGMS99,FKSV99,HRR98,KN97]. We seek protocols in which the total communication and the number of rounds are small. The parties should be able to compute their protocol responses quickly, using little storage space. Ideally, we desire that only one round of the protocol need involve raw input and that each party can compute her message for this round in just a single pass over her input, with little space and little time required to process each item. In theory, this allows the parties to regard their inputs as unbuffered data feeds that need not be stored at all, *i.e.*, the inputs may be regarded as *data streams* (cf. [HRR98]).

3 Sublinear Private Approximation of the Hamming Distance

In this section we obtain a private two-party protocol for the approximate Hamming distance between two strings. Such a protocol allows Alice, holding an input $a \in \{0,1\}^n$, and Bob, holding $b \in \{0,1\}^n$, to learn an ϵ-approximation of the Hamming distance between a, b with negligible failure probability δ, without learning additional information on the other player's input except what follows from the Hamming distance between the inputs. Our protocol uses $\tilde{O}\left(n^{1/2}/\epsilon\right)$ communication and 3 rounds of interaction. By contrast, an exact computation of the Hamming distance, private or not, requires $\Omega(n)$ communication [KN97].

Notation and conventions. We let $d_h(a,b)$ denote the Hamming distance between a, b, and $w_h(x)$ denote the Hamming weight of an n-bit string x. The asymptotic notation $\tilde{O}(x)$ should always be read as "$x \cdot \log^{O(1)} n$". In the following, when referring to an approximation we will often omit the parameter δ. In such a case δ should be understood to be either a constant (say $1/4$) or negligible ($n^{-\omega(1)}$), as will be made clear from the context. Note that a small constant failure probability δ can always be decreased to a negligible one by $\tilde{O}(1)$ repetitions. Finally, k will denote a security parameter, which may be replaced either by n^γ for an arbitrarily small γ or by $\tilde{O}(1)$, depending on the strength of the cryptographic assumptions being made.

3.1 The Sketching Approach to (Non-Private) Approximations

Before describing our private protocol it is instructive to consider the non-private variant of the problem. We first survey known efficient solutions for this problem, then we explain why a naive attempt to make these solutions private fails.

There are several known methods for approximating the Hamming distance using poly-logarithmic communication [AMS96,KOR98,FKSV99,CPSV00,KN97]. More specifically, the best $\langle \epsilon, \delta \rangle$-approximations require $O(\log n \log(1/\delta)/\epsilon^2)$ communication. These methods can all be viewed as based on the following *sketching* approach.

Definition 5. *A* sketching protocol *for a 2-argument function $f : \{0,1\}^n \times \{0,1\}^n \to \mathbb{N}$ is defined by:*

– *A* sketching function, $S : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$ *mapping one input and a random string to a* sketch *consisting of a (presumably short) string.*
– *A (deterministic)* reconstruction function $G : \{0,1\}^m \times \{0,1\}^m \to \mathbb{R}$, *mapping a pair of sketches to an output.*

On inputs a, b, the protocol proceeds as follows. First, Alice and Bob locally compute a sketch $s_A = S(a, r)$ and $s_B = S(b, r)$ respectively, where r is a common *random input. Then, the parties exchange sketches, and both locally output $g = G(s_A, s_B)$. We denote by $g(a, b)$ the* randomized function *defined as the output of the protocol on inputs a, b. A sketching protocol $\langle \epsilon, \delta \rangle$-approximates f if g $\langle \epsilon, \delta \rangle$-approximated f.*

Note that in the public random string model, the communication complexity of a sketching protocol as above is of the order of the sketch size. In some contexts, the use of common randomness can be eliminated at a moderate cost (cf. [New91]). In cryptographic protocols, the length of a shared random string can be taken very small and expanded to the required length by each party using a pseudorandom number generator.

We briefly review a simple efficient sketching protocol for the Hamming distance [KOR98,CPSV00].

Example 1. **(A sketching protocol for the Hamming distance.)** Let the common random input define a 0/1-valued matrix R, with $\log n$ rows and n columns, in which each entry of the i-th row (independently) takes the value 1 with probability $p_i = \beta^i$ for some constant β depending on ϵ. The sketching function is defined by $S(x, R) = Rx$, where R, x are viewed as a matrix and a vector over GF(2). From the sketches Ra,Rb the distance $d_h(a, b)$ can be approximated. (Roughly, the distance is estimated by observing that $(Ra)_i = (Rb)_i$ with probability close to 1/2 if $d_h(a, b) \gg n\beta^i$ and with probability 1 if $d_h(a, b) \ll n\beta^i$. The critical i can be found by sampling from repeated trials. The entire protocol can be repeated to reduce the distortion and failure probability.) The communication complexity of this sketching protocol is $O(\log n)$.

3.2 Achieving Privacy

Our goal is to obtain a sublinear-communication *private* approximation protocol for the Hamming distance function. A natural approach for achieving this goal is to find a *general* way for converting an efficient sketching protocol approximating a function f into a private protocol approximating f.

Suppose that the (randomized) function g produced by the sketching protocol is functionally private with respect to f. This is indeed the case for the sketching protocol from Example 1 as well as for the other sketching protocols for the Hamming distance proposed in the literature. Then, to approximate f privately, it suffices to let the players privately compute g.

While a general-purpose private computation protocol can be used to evaluate g, its communication complexity will be (at least) linear in n, whereas we would like to obtain a sublinear-communication private protocol for g. At first glance, the following naive protocol seems to work. Let the players exchange a common random input r (or a succinct representation of a *pseudo-random* input r), and let each player locally compute its sketch based on its input and r. Then, apply a general-purpose private computation protocol to evaluate $g = G(s_A, s_B)$ from the sketches s_A, s_B. The communication complexity of privately computing G can be made linear in the circuit size of G times the security parameter [Yao86,GMW87], and so if the sketches are short and G is not too complex the entire protocol can be implemented with sublinear communication. However, this naive protocol generally fails to be private. Indeed, even when g is functionally private with respect to f, knowing the output of g *together with the random input* r (which was used to generate this output) can reveal additional information on the inputs. For instance, in the sketching protocol of Example 1, Alice can deduce Rb from her input a, the output $R(a - b)$ and the common random input R. It is not hard to see that based on a and $d_h(a, b)$ alone, it is impossible to generate R, r such that R is distributed as in Example 1 and $Rb = r$ holds with overwhelming probability. (For instance, given that $a = 0$, $b = e_i$, and $d_h(a, b) = 1$, r should be equal to the i-th column of R, which is impossible to guess with high probability from a and $d_h(a, b)$ alone.) Thus, the view of Alice is not simulatable in the ideal model.

Unfortunately, we do not know whether the sketching method of Example 1 can be made private with sublinear communication, nor can we obtain a private protocol from any other efficient protocol for approximating the Hamming distance appearing in the literature. While these may still be useful in other settings for privately computing the Hamming distance (see Appendix B.2), we will have to rely on alternative methods for solving our problem.

Underlying our solution is a combination of two different sketching protocols with small communication overhead (also referred to as *estimators*) for the Hamming distance. For both these estimators, the produced reconstruction functions are functionally private, and we show how to privately implement them with low communication overhead. The first estimator, based on sampling, is efficient when the distance is high. Towards a private implementation of this estimator, we devise a special-purpose private protocol for comparing the bits in a random location, which may be of independent interest. This protocol relies on the use of a sublinear-communication $\binom{n}{1}$-OT protocol[5] [KO97,GIKM00,NP99,CMS99]. The second estimator, for which we give several constructions, is efficient when the distance is low, and in fact produces an *exact* result in this case. In the following sections we describe each of the two private estimators, and then combine them to obtain the final protocol, which works (privately and efficiently) for any distance.

3.3 The High Distance Estimator

Suppose that $d = d_h(a, b)$ is guaranteed to be larger than some threshold $d_{\min}$ (which will be specified later). If $d_{\min}$ is large, then Alice and Bob can efficiently approximate d by randomly sampling a small number of bits in matching positions from their inputs. This may be viewed as the simplest form of sketching: the random input includes several random indices, the sketch contains the bits indexed by the random input, and the output is obtained by scaling the relative distance between the sketches. Specifically, Alice and Bob count the number of differences Δ in $s = O(\frac{n}{\epsilon^2 d_{\min}})$ randomly selected matching bits of their inputs and compute the estimate $g = \frac{\Delta \cdot n}{s}$. By a Chernoff-bound argument, g is an ϵ-approximation of d. Note that the mod-2 sum of bits in a pair of randomly selected matching positions is functionally private with respect to $d_h(a, b)$ and the functional privacy of the protocol's output follows.

We will show that for this particular sketching protocol, the output function g can be *privately* computed with a very small communication complexity. Our main building block is a private protocol for comparing a randomly sampled pair of bits. Formally, this protocol computes the randomized function Sample-XOR defined as

$$\text{Sample-XOR}(a, b) = (a_r \oplus b_r, a_r \oplus b_r) \quad \text{where } r \stackrel{\text{R}}{\leftarrow} [n]$$

Recall that in a private computation (of either a deterministic or a randomized function) it should be possible to simulate the view of each party based on its own input and output, but without access to the other party's input. In particular, a protocol for Sample-XOR should keep the choice of r private from each party, since the output reveals no information on the *location* of the sampled pair of bits.

Figure 1 describes a private protocol for the function Sample-XOR. Our protocol uses $\binom{n}{1}$-OT as a subprotocol. The proof of Lemma 1 appears in Appendix C.2.

Lemma 1. Private-Sample-XOR *is a private protocol computing the randomized function* Sample-XOR.

Efficiency. $\binom{n}{1}$-OT protocols with sublinear communication can be constructed from private information retrieval (PIR) protocols [CGKS95,KO97,CMS99] using a reduction from [NP99]. Under a specific number-theoretic intractability assumption called the *Φ-Hiding Assumption* [CMS99], there exists a 2-round PIR protocol with $\tilde{O}(1)$ communication. Under the (more general) assumption that homomorphic public-key cryptosystems exist, $O(n^\gamma)$ communication is achieved for arbitrarily small constant $\gamma > 0$ [KO97,Man98,Ste98]. It follows that Protocol Private-Sample-XOR can be implemented with 3 rounds and with the same asymptotic communication complexity as above.

Given approximation parameters ϵ, δ, our private sampling estimator for the high distance case will be implemented using $s = O((n/d_{\min}) \cdot \log(1/\delta)/\epsilon^2)$ parallel invocations of Protocol Private-Sample-XOR. As argued above, when $d = d_h(a, b) \geq d_{\min}$ then an $\langle \epsilon, \delta \rangle$-approximation of d can be computed from the s outputs by multiplying their sum by n/s. Regardless of the distance between the inputs a, b, the view of each party in these invocations can be simulated from its input and $d_h(a, b)$, which is sufficient to guarantee that the protocol is indeed private with respect to d_h. Summarizing, we have:

[5] $\binom{n}{1}$-OT ("n choose 1 Oblivious Transfer") is a private 2-party protocol for the following function: the input of the *sender* is an n-bit string x and the input of the *receiver* is an index $i \in [n]$; the output of the receiver is the bit x_i, and the sender has no output. Note that in the current context we only require security against a *passive* adversary.

Private-Sample-XOR

1. Alice picks a random mask $m_A \stackrel{R}{\leftarrow} \{0,1\}$ and a random shift amount $r_A \stackrel{R}{\leftarrow} [n]$.
   She computes the n-bit string $a' \stackrel{\text{def}}{=} (a \ll r_A) \oplus m_A$
   (where for any $x \in \{0,1\}^n, r \in [n]$ and $m \in \{0,1\}$, we denote by $x \ll r$ a cyclic shift of x by r bits to the left, and by $x \oplus m$ the string whose i-th bit is $x_i \oplus m$).
   Symmetrically, Bob picks $m_B \stackrel{R}{\leftarrow} \{0,1\}$ and $r_B \stackrel{R}{\leftarrow} [n]$, and computes $b' \stackrel{\text{def}}{=} (b \ll r_B) \oplus m_B$.
2. Alice and Bob invoke in parallel two $\binom{n}{1}$-OT protocols:
   - Alice retrieves $z_A \stackrel{\text{def}}{=} b'_{r_A}$ from Bob;
   - Bob retrieves $z_B \stackrel{\text{def}}{=} a'_{r_B}$ from Alice.
3. Alice sends $z'_A \stackrel{\text{def}}{=} z_A \oplus m_A$ to Bob. In parallel Bob sends $z'_B \stackrel{\text{def}}{=} z_B \oplus m_B$ to Alice.
   Both parties locally output $z'_A \oplus z'_B$.

Fig. 1. A private protocol for the function Sample-XOR

Lemma 2. (Private approximation for the high distance case.) *Let $\mathcal{OT}$ be an arbitrary $\binom{n}{1}$-OT protocol (with security against a passive adversary and a security parameter k). Then, there exists a protocol for approximating $d_h(a,b)$ whose communication complexity is $\tilde{O}((n/d_{\min})/\epsilon^2)$ times that of $\mathcal{OT}$, and whose round complexity is that of $\mathcal{OT}$ plus 1, such that:*

- *The protocol is* private *with respect to the function $d_h(a,b)$;*
- *If $d = d_h(a,b) \geq d_{\min}$, the protocol outputs an ϵ-approximation of d with overwhelming probability.*

In particular, under the Φ-Hiding Assumption there exists a 3-round protocol as above with $\tilde{O}((n/d_{\min})/\epsilon^2)$ communication. Note that the messages for all the invocations of $\binom{n}{1}$-OT can be computed incrementally in parallel, with a single pass over the raw data, and with storage proportional to the communication complexity.

The sampling estimator will not give a reliable estimate when the distance d is significantly smaller than $d_{\min}$ because its variance will be too high (that is, it is likely that no differences will be detected). It will, however, reliably indicate that d is small. In that case, Alice and Bob will know that they should use an estimator for the low distance case, described in the next section.

3.4 The Low Distance Estimator

We describe two private protocols for the low distance case, i.e. the case that $d \leq d_{\max}$ for some threshold $d_{\max}$ to be later specified. Each of the private protocols is based on a (different) sketching protocol having the following properties:

- The induced function g is *almost determined* by d_h. That is, except with negligible probability, $g(a,b)$ takes a specific value determined by $d_h(a,b)$. In particular, g is functionally private with respect to d_h.
- If $d_h(a,b) \leq d_{\max}$, then $g(a,b) = d_h(a,b)$ with overwhelming probability.

It is important to note that for any sketching protocol satisfying the first property above, a private computation of g may proceed according to the naive approach described in Section 3.2. That is, the parties may exchange a common random input in the clear, then locally compute the sketches based on their inputs and the random input, and finally apply a (general-purpose) private computation protocol for evaluating the reconstruction function G on their sketches. (Intuitively, in this case the common random input r gives almost no information about the inputs except what follows from g; formally, a simulator may sample r independently of g, compute a sketch s, and then invoke the simulator for the private computation of G.) However, for such a protocol to be communication-efficient, it is important that G can be computed by a small circuit, preferably linear or nearly-linear in the sketch size.

Below we describe a sketching protocol based on hashing. Its description is self-contained, and it only requires the private computation of a very simple reconstruction function. In Appendix B.1, we give a second protocol based on Reed-Solomon codes. Its reconstruction function G will be quite complex, and

consequently its private implementation will require a heavy use of generic private computation (yet its asymptotic efficiency will be good enough for our purposes).

A protocol based on hashing. We first describe this protocol under the assumption that the two parties have access to a large common source of randomness, which in particular defines several independent (2-universal) hash-functions. This assumption will be dispensed with later. Given a security parameter k, the sketch of an input $x \in \{0,1\}^n$ is computed as follows:

1. Randomly partition the n bits of x into $d_{\max}$ buckets. (With probability $1 - 2^{-\Omega(k)}$ no bucket gets more than $k \log n \cdot (n/d_{\max})$ bits, or more than $k \log n$ bits in which a, b differ.)
2. For each of the $d_{\max}$ buckets, further partition its bits into $(k \log n)^2$ sub-buckets. (Now, if a given bucket contains $k \log n$ differences, then each of its sub-buckets will contain at most one difference with constant probability.) Repeat this procedure k independent times, and let B_{ijh} denote the contents of the j-th sub-bucket of the i-th bucket in the h-th invocation (where $1 \leq i \leq d_{\max}$, $1 \leq j \leq (k \log n)^2$, $1 \leq h \leq k$).
3. Hash the contents of each sub-bucket B_{ijh} to a k-bit string β_{ijh}.

The sketch of a string x will consist of all $d_{\max} \cdot k^3 \log^2 n$ strings β_{ijh} obtained via the above process.

Let $(\beta_{ijh}(a), \beta_{ijh}(b))$ denote the (correlated) values of β_{ijh} when the above process is applied on inputs a, b using the same random input.

Lemma 3. *Suppose that $d_h(a,b) \leq d_{\max}$. Then, with probability $1 - 2^{-\Omega(k)} \cdot d_{\max}$,*

$$d_h(a,b) = \sum_{i=1}^{d_{\max}} \max_{1 \leq h \leq k} \left| \left\{ 1 \leq j \leq (k \log n)^2 \ : \ \beta_{ijh}(a) \neq \beta_{ijh}(b) \right\} \right| . \tag{1}$$

Proof (sketch): As noted in the description of the sketching function, each of the k attempts of secondary hashing succeeds with a constant probability to isolate *all* of the bit differences mapped to its bucket. Hence, with probability $1 - 2^{-\Omega(k)}$ at least one of them succeeds. Moreover, for any instance i, j, h, the probability of the third-level hashing mapping distinct values $B_{ijh}(a), B_{ijh}(b)$ to the same k-bit string is $2^{-\Omega(k)}$. The claim follows by a union-bound argument. ∎

Suppose that the reconstruction function of the sketching protocol is defined by the right hand side of Eq. (1). By symmetry, the output g is already functionally private. But, because g fails to be almost determined by d_h over the entire range of inputs, the naive private implementation discussed in Section 3.2 cannot be used, as explained there. In this case, however, a very simple modification to the reconstruction function can fix this situation. The modified reconstruction will first compute an estimate $\tilde{d}$ by applying the right hand side of Eq. (1) to the sketches, and then it will output $\tilde{d}$ if $\tilde{d} \leq d_{\max}$ and output "fail" otherwise. It is easy to verify that: (1) the modified reconstruction function can be computed by a circuit of size $\tilde{O}(d_{\max})$; (2) if $d = d_h(a,b) \leq d_{\max}$, then $g(a,b) = d$ with overwhelming probability; (3) if $d > d_{\max}$ then $g(a,b)$ outputs "fail" with overwhelming probability. Finally, note that, using linear hashing, the hash of each sub-bucket β_{ijh} can be computed incrementally, requiring little space and a single pass over the data.

Based either on this hashing-based sketching protocol for the low distance case or on the sketching protocol of Appendix B.1, a private protocol for the low distance case may be constructed as outlined in the beginning of this section. That is, in the first round Alice sends to Bob a seed to a pseudo-random generator which will be used to produce the required common randomness. Then, each party locally computes the sketch of its input, and together they apply a general-purpose private circuit evaluation protocol to evaluate the reconstruction function on their sketches. Using Yao's protocol [Yao86], this private computation requires only two additional rounds. Its communication complexity is of the order of the security parameter times the size of a circuit computing G. Summarizing, we have:

Lemma 4. (Private approximation for the low distance case.) *Suppose that trapdoor permutations exist. Then, for any $1 \leq d_{\max} \leq n$, there exists a 3-round protocol with $\tilde{O}(k \cdot d_{\max})$ communication, such that:*

- *The protocol is* private *with respect to the function $d_h(a,b)$;*
- *If $d = d_h(a,b) \leq d_{\max}$, the protocol outputs the* exact *value of d with overwhelming probability;*
- *If $d = d_h(a,b) > d_{\max}$, the protocol outputs "fail" with overwhelming probability.*

3.5 The Combined Protocol

Using the protocols from Lemma 2 and Lemma 4 as subprotocols, our final protocol proceeds as follows. Given the desired approximation quality ϵ and a security parameter k:

1. Invoke the (high distance) protocol of Lemma 2 with parameters k, ϵ, and $d_{\min} = n^{1/2}/\epsilon$. Let d_1 denote its output.
2. In parallel, invoke the (low distance) protocol of Lemma 4 with parameters k and $d_{\max} = n^{1/2}/\epsilon$. Let d_2 denote its output.
3. If $d_2 =$"fail", output d_1; else output d_2.

The privacy of the combined protocol follows from the privacy of each of its two subprotocols over the entire range of inputs. Its correctness follows from the facts that: (1) if $d > d_{\max}$ then (with overwhelming probability) the final output is produced by the high distance subprotocol, and, since $d > d_{\max} \geq d_{\min}$ this output is ϵ-correct; (2) if $d \leq d_{\max}$, then the final output is produced by the low distance subprotocol, which is guaranteed in this case to be correct with overwhelming probability. The combined protocol also requires only a single pass over the raw data, and only small storage complexity, proportional to the communication.

Substituting the complexity parameters of the two subprotocols yields the following theorem:

Theorem 1. *Suppose that the Φ-Hiding Assumption holds (respectively, homomorphic public-key encryption exists). Then, the Hamming distance function can be privately ϵ-approximated with communication complexity $\tilde{O}(n^{1/2}/\epsilon)$ (respectively, $O(n^{1/2+\gamma}/\epsilon)$ for any constant $\gamma > 0$) and 3 rounds of interaction.*

In Appendix B.2, we show that it is possible to obtain improved efficiency if a linear amount of free offline communication is allowed before Alice and Bob receive their inputs.

4 Efficient Approximations of #P-hard Functions

We now turn our attention to privately approximating natural #P-hard problems, where the goal is to achieve *polynomial time* private approximations. This is in contrast to problems on massive data sets that we have been focusing on thus far, where polynomial time exact private computation is possible, and the goal is to achieve lower complexity (sublinear in the Hamming distance case).

We start by observing that artificially constructing #P-hard problems which can be privately approximated is straightforward. For example, consider any #P-hard problem $f(x)$ with output in the range $[0, 2^n]$. Then $g(x) = f(x) + 2^{2n}$ is computationally equivalent to $f(x)$, and, in particular, is computationally "interesting" iff f is. Although, for many values of ϵ, 2^{2n} is a $(1 \pm \epsilon)$-factor private approximation to $g(x)$, this approximation doesn't approximate any interesting quantity. Thus, in general, while some exact #P-hard problem may be interesting, its approximate version may not be.

In this section, however, we give private approximations to *natural* #P-hard problems, most notably the permanent. Below we provide some motivation for this problem in our context, and sketch our approximation solution.

The Permanent and Some Applications. The number of perfect matchings in a bipartite graph between two sets of n vertices is equal to the permanent (over the integers) of the graph's adjacency matrix. Counting the number of perfect matchings is a #P-hard problem. As one might expect of #P-hard problems, the permanent has applications to a wide variety of counting problems, including some that arise naturally in physics. Less obvious (but true nevertheless) is that many natural problems reduce to the permanent *in an approximation-preserving way*, implying that a private approximation to the permanent immediately yields private approximations to these problems. For example, the number of tilings of certain lattices can easily be expressed as a permanent, so that an approximation to the permanent gives an approximate count of the number of tilings. There are also problems concerning bond strength in molecules that reduce to the permanent. For example, consider a benzoid structure, which can be represented as a connected graph whose vertices are labeled carbon or hydrogen, subject to the following conditions (See Figure 2):

- the graph is a subgraph of the hexagonal lattice (a bipartite graph)
- the hydrogen atoms each have degree one (and are irrelevant)

– each carbon atom has degree 4, is adjacent to 2 or 3 other carbon atoms, and, in each configuration, has single or double bonds to each adjacent carbon atom such that the double bonds form a perfect matching of the carbons.

The Pauling bond order of a particular C-C bond in the molecule is the expected bond order of that bond in a randomly-chosen configuration. Thus the Pauling bond order is given by one plus the probability that the edge corresponding to the bond occurs in a randomly chosen perfect matching in the graph corresponding to the carbon atoms in the molecule. This is a theoretical prediction of the physical "strength" of a bond, *e.g.*, the dissociation energy, that is even harder than the bond order to compute, and depends on things like the shapes of orbits.

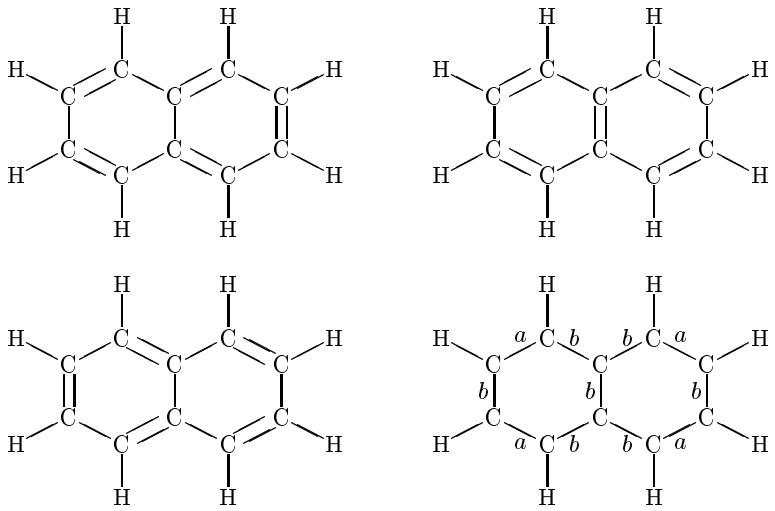

Fig. 2. The first three diagrams depict configurations of single and double bonds in naphthalene, $C_{10}H_8$, which is used to repel moths. The double bonds form a perfect matching of the carbon skeleton. The order of a labeled bond in the molecule is its expected order of that bond in a configuration chosen uniformly at random from the three possibilities, *i.e.*, the order of each bond is one plus the probability that it occurs in a uniformly random perfect matching. Thus, the bonds marked a in the fourth diagram are of order 5/3 and those marked b there are of order 4/3.

Approximating the Permanent. In their seminal work [JS89], recently updated [JSV00], Jerrum, Sinclair and Vigoda gave an $\langle \epsilon, \delta \rangle$-approximation for the permanent computable in polynomial time. We use a modification of their technique to achieve a private approximation.

Theorem 2. *Suppose two players share (in any reasonable way) an $n \times n$ matrix M with non-negative entries. There is a polynomial time protocol whereby the players can privately compute an $\langle \epsilon, \delta \rangle$-approximation to the permanent of M.*

Proof (sketch): We provide here a very rough sketch of the proof, with a more technically detailed proof in Appendix C.3.

We begin by noting that any efficiently-computable randomized function admits an efficient private protocol [Yao82,GMW87]. Thus it suffices to show how to efficiently approximate the permanent in a functionally private way. To this end, we use ideas of [JSV00].

The technique [JSV00] uses a rapidly-mixing Markov chain to show how to sample from the set of all perfect matchings on a graph from a distribution that is statistically indistinguishable from uniform. This is then used to approximate the probability p_e that a certain edge e is in a random matching (roughly, by sampling random matchings and checking the fraction of them that contains e). Next the probability

$p_{e_2|e_1}$ that e_2 is in a random matching that contains e_1 is approximated, and so on, up to the probability $p_{e_n|e_1,\ldots,e_{n-1}}$ for an edge e_n to be in a random matching, given that $e_1, \ldots, e_{n-1}$ are in the matching. This is done for a sequence of edges $e_1, ..., e_n$ such that each of the above probabilities is at least $1/n$, so that, in particular, $\{e_1, \ldots, e_n\}$ is a perfect matching (an appropriate sequence of edges is itself found using Markov chain sampling). Now, the number of perfect matchings can be written as

$$1/(p_{e_1} p_{e_2|e_1} \cdots p_{e_n|e_1,\ldots,e_{n-1}}).$$

Since an approximation for each probability can be found, a (non-private) approximation for the product, and thus the permanent, follows.

In order to make the approximation private, we first note that the approximation for each individual probability $p_{e|.}$ is already functionally private with respect to $p_{e|.}$, *i.e.*, does not leak additional information about the graph. However, the product of approximations *does* potentially leak information about its factors (*e.g.*, the standard deviation of the product approximation depends on the factors), and thus the product of approximations is not functionally private.

Fortunately, we are able to show how to avoid this leakage, by manipulating success probabilities in various ways. One important process we call *enriching*: Given a coin with success probability p, whose value is unknown except that $p \ll 1/t$ for some real number $t \geq 1$, we can enrich the coin to have success probability tp. Besides enriching, we will also use other constructions on success probabilities: Given coins with success probabilities p, q and r, we can form coins with success probabilities pq, $1-p$, and $rp+(1-r)q$.

We proceed, roughly, as follows. If the security parameter k and n are at least exponentially far apart, we show that there's a trivial algorithm for privately approximating the permanent. Otherwise, we use the tools described above to construct coins with success probability $p'_{e|.} = p_{e|.}^{1/n}$ for each probability $p_{e|.}$, correct to within $\left(1 \pm 2^{-k}/n\right)$, using its $O(k)$-term Taylor expansion. The joint success of these coins has success probability $M^{-1/n}(1 \pm 2^{-k})$, *i.e.*, indistinguishable from $M^{-1/n}$, where M is the value of the permanent. Furthermore, $M^{-1/n} \geq 1/n$, so $M^{-1/n}(1 \pm \epsilon/n)$ can be estimated by sampling, using Lemma 5. It follows that the $-n$ power is $M(1 \pm O(\epsilon))$. ∎

Remarks:

• From the above proof we can conclude that p_e, the probability that a given edge e appears in a random perfect matching in a graph, is privately approximable with good relative error, provided the probability is large enough (at least a negative power of n), and is privately approximable with good additive error in any case. This follows directly from using the [JSV00] sampling techniques for approximating p_e. From the discussion in the proof, it is not difficult to see that this is a #P-hard problem, since it is reducible from the permanent.

• We note that the techniques used in the above proof may be applicable to approximate a more general class of problems than just the permanent. Indeed, the technique of rapidly-mixing Markov chains is inherently suited for use in functionally-private approximations, since by definition of "rapidly mixing," the Markov chain supports sampling from a distribution of items that is statistically indistinguishable from uniform. If we then sample to estimate the fraction of items satisfying some property, the resulting estimate will depend only on the fraction, not otherwise on the set of items or the input used to generate them. Often, as in the case of the permanent, we don't want to estimate the fraction of objects satisfying some property, but rather some function of several such fractions (such as their product). To this end, our techniques of manipulating probabilities, and using j-th roots (through a Taylor expansion estimation) seem to be useful to compute more general functions. We provide more details in Appendix C.4.

5 Conclusion

We have given formal definitions and techniques for secure and private approximate multiparty computations. The results of Section 3 require a general-purpose private computation on inputs of size $\tilde{O}(\sqrt{n})$ (resulting in $\tilde{O}(\sqrt{n})$ communication). While this is certainly better than a private computation on input of size n, the result may still be too expensive to be considered practical. It remains open to improve these results—perhaps to achieve $\log^{O(1)}(n)$ communication, as is possible in the insecure setting [AGMS99]. Nonetheless, we think our work represents the important step of introducing and formally understanding private approximations, as well as a first step towards practical solutions.

Acknowledgements

We thank Dana Randall for suggesting the benzoid application in Section 4. We are grateful to Jessica Fong for helpful discussions and collaboration in early stages of this work.

References

[AR00] R. Agrawal and S. Ramakrishnan. Privacy-preserving data mining. In *Proceedings of of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 439–450, 2000.

[AGMS99] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy, Tracking Join and Self-Join Sizes in Limited Storage. In *Proceedings of the 18th Symposium on Principles of Database Systems* (PODS), pp. 10–20, ACM Press, New York, 1999.

[AMS96] N. Alon, Y. Matias, and M. Szegedy, The Space Complexity of Approximating the Frequency Moments. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (STOC), pp. 20–29, May 1996.

[Bea91] D. Beaver, "Foundations of Secure Interactive Computing", *CRYPTO '91, Lecture Notes in Computer Science (LNCS) 576,* Springer-Verlag, 1991.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson, Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computing. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (STOC), pp. 1–10, 1988.

[CMS99] C. Cachin, S. Micali, and M. Stadler, Computationally Private Information Retrieval with Polylogarithmic Communication. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of Lecture Notes in Computer Science, pp. 402–414, Springer-Verlag, 1999.

[Can00] R. Canetti, Security and Composition of Multiparty Cryptographic Protocols, *Journal of Cryptology,* Vol. 13, No. 1, Winter 2000.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård, Multiparty Unconditionally Secure Protocols. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (STOC), pp. 11–19, 1988.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the 36th Annual IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version: *J. of the ACM*, 45:965–981, 1998.

[CPSV00] G. Cormode, M. Paterson, S. Sahinalp, and U. Vishkin, Communication complexity of document exchange. In Proceedings of the 11th ACM-SIAM Annual Symposium on Discrete Algorithms, pages 197–206, San Francisco, CA, 2000.

[DIM97] *DIMACS Special Year on Massive Data Sets, 1997–1999,* `http://dimacs.rutgers.edu/SpecialYears/1997_1998/`.

[DIM00] *DIMACS Special Focus on Data Analysis and Mining, 2001–2004,* `http://dimacs.rutgers.edu/SpecialYears/2001_Data/`.

[FFSW00] J. Feigenbaum, J. Fong, M. Strauss, and R. Wright, Secure Multiparty Computation of Approximations, presented at *DIMACS Workshop on Cryptography and Intractability*, March 20–22, 2000.

[FKSV99] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, An Approximate L^1-Difference Algorithm for Massive Data Streams. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), pp. 501–511, October 1999.

[FS00] J. Fong and M. Strauss, An approximate L^p-difference algorithm for massive data streams. In *Proceedings of 17th Symposium on Theoretical Aspects of Computer Science* (STACS), pp. 193–204, February, 2000.

[GIKM00] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. of Computer and System Sciences*, 60(3):592–629, 2000. A preliminary version appeared in *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing* (STOC), 1998.

[Gol98] O. Goldreich, *Secure Multi-Party Computation,* (Working Draft, Version 1.1). September 1998. Available from `http://philby.ucsd.edu/cryptolib/BOOKS/oded-sc.html`.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, How to Play Any Mental Game—A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (STOC), pp. 218–229, 1987.

[HKKN01] S. Halevi, E. Kushilevitz, R. Krauthgamer, and K. Nissim, Private approximations of NP-hard functions. To appear, STOC 2001.

[HRR98] M. Rauch Henzinger, P. Raghavan, and S. Rajagopalan. *Computing on data streams.* Technical Report 1998-011, Digital Equipment Corporation Systems Research Center, May 1998.

[Ind00] P. Indyk, Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2000.

[JS89] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing* **18** (1989), 1149–1178.

[JSV00] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. Electronic Colloquium on Computational Complexity, Report no. 79, 2000.

[KN97] E. Kushilevitz and N. Nisan, *Communication Complexity*. Cambridge University Press, 1997.

[KO97] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th Annual IEEE Symp. on Foundations of Computer Science* (FOCS), pages 364–373, 1997.

[KOR98] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (STOC), pp. 614–623, May 1998.

[LP00] Y. Lindell and B. Pinkas, Privacy preserving data mining. In *Advances in Cryptology - CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54. Springer-Verlag, 2000.

[Man98] E. Mann. Private access to distributed information. Master's thesis, Technion - Israel Institute of Technology, Haifa, 1998.

[Minc82] H. Minc. Permanents. In *Encyclopedia of Mathematics and its Applications* **6**, Addison-Wesley, 1982.

[MR91] S. Micali and P. Rogaway, "Secure Computation", unpublished manuscript, 1992. Preliminary version in *CRYPTO '91, LNCS 576,* Springer-Verlag, 1991.

[NN01] M. Naor, and K. Nissim. Communication Preserving Protocols for Secure Function Evaluation. To appear, STOC 2001.

[New91] Ilan Newman, Private vs. Common Random Bits in Communication. Complexity. In Information Processing Letters 39(2): 67-71, 1991.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proc. of 22th STOC*, pages 213–223, 1990.

[NP99] M. Naor and B. Pinkas, Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing* (STOC), pp. 245–254, May 1999.

[Ste98] J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371. Springer-Verlag, 1998.

[Yao82] A. C. Yao, Protocols for Secure Computation. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science* (FOCS), pp. 160–164, November 1982.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th Annual IEEE Symp. on Foundations of Computer Science*, pages 162–167, 1986.

A Further discussion of secure approximation protocols

A.1 Discussion of Definition 3

Intuitively, leakage to P_i from x_i and $\hat{f}_i(\mathbf{x})$ could actually take three forms: P_i learns extra information about P_j's input x_j, or about $f_j(\mathbf{x})$, or about $\hat{f}_j(\mathbf{x})$. However, in our restricted setting, only the first kind of leakage might happen. This allows the simulator S_i in Definition 3 to produce a correctly distributed $\hat{f}_i(x)$ without regard to the distribution of other players' inputs. In the more general setting, S_i would have to produce $\hat{f}_i(x)$ both correctly distributed and properly correlated with other players' outputs.

More formally, recall that the definitions for privacy of a multiparty protocol require, intuitively, that party i's view of the protocol can be simulated *including the proper correlation with the other players' outputs*. (See [Gol98,Can00] for these definitions and more discussion.) That is, for two parties Alice and Bob computing f using protocol π, there are simulators S_1 and S_2 such that

$$\{(S_1(x_1, f_1(\mathbf{x})), f_2(\mathbf{x})\}_{x_i \in \{0,1\}^*} \stackrel{c}{\equiv} \{(\mathrm{view}_1(\mathrm{ex}_\pi(\mathbf{x})), \mathrm{out}_2(\mathrm{ex}_\pi(\mathbf{x}))\}_{x_i \in \{0,1\}^*}$$

and, similarly, with the players' roles reversed, where $\mathrm{ex}_\pi(\mathbf{x})$ denotes the random variable consisting of all possible executions of the protocol π in which the input vector is $\mathbf{x}$, over all possible random coin flips of the players. Given an execution e of π, we define $\mathrm{view}_i(e)$ and $\mathrm{out}_i(e)$ to be the view and output of P_i.

The simpler definition,

$$S_1(x_1, f_1(\mathbf{x})) \stackrel{c}{\equiv} (\mathrm{view}_1(\mathrm{ex}_\pi(\mathbf{x}))$$

will *not* in general work, since, under that definition, there would not necessarily be simulators for protocols in which Alice's view is correlated with Bob's input.

The analogous definition of functional privacy, taking into account correlation, would be that there exists a simulator S_1 such that

$$(S_1(x_1, f_1(\mathbf{x})), \hat{f}_2(\mathbf{x})) \stackrel{c}{\equiv} (\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x})),$$

and, similarly, with the players' roles reversed. We now argue that our simpler Definition 3 is appropriate for subset-learning multi-output functions $\hat{f}$ based on single-output functions ζ, both in the independent-output and identical-output cases. Assume the computation is a two-party computation in which both players, Alice and Bob, receive output.

If the outputs $\hat{f}_1(\mathbf{x})$ and $\hat{f}_2(\mathbf{x})$ are independent, then our definition immediately implies

$$(S_1(x_1, f_1(\mathbf{x})), \hat{f}_2(\mathbf{x})) \stackrel{c}{\equiv} (\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x})),$$

where the three occurrences of ζ, as $\hat{f}_1$ or $\hat{f}_2$, are independent. On the other hand, in the identical output case, there is indeed (perfect) correlation between Alice's and Bob's outputs. In this case, our definition implies that there is a simulator S' for Alice such that

$$S'_1(x_1, f(\mathbf{x})) \stackrel{c}{\equiv} (\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x})),$$

i.e., Alice learns nothing from *both* approximate outputs that she can't learn from her own input and the exact output.

Our simpler definition also works in the m-output case provided the outputs are independent.

A weaker definition.

We now consider an alternate, generally weaker definition.

Definition 6. *Let $f(\mathbf{x})$ be a deterministic function, and let π be a protocol in which exactly one of the players, P_1, gets a randomized output, $\hat{f}(\mathbf{x})$. We say that π is a* weakly private approximation protocol for *f if:*

- **(Correctness)** *The output $\hat{f}(\mathbf{x})$ is an $\langle \epsilon, \delta \rangle$-approximation to $f(\mathbf{x})$.*
- **(Privacy)** *For each player i, there is a simulator S_i for P_i such that*

$$S_1(x_1, f_1(\mathbf{x})) \stackrel{c}{\equiv} (\mathrm{view}_1(\mathrm{ex}_\pi(\mathbf{x}))$$

and similarly for the other players.

Note that, since f is deterministic, each player's view is independent of f. Intuitively, this definition says (at least) that no player's view reveals anything about other players' inputs and exact outputs not implied her own input and exact output. It is possible that some player P_i, $i \neq 1$, learns something about P_1's approximate output not implied by her own input (and null output).

Under this definition, the 3-party protocol Hamming-3 of Figure 3 for the Hamming distance

$$(\langle a_i \rangle, \langle b_i \rangle, \perp) \rightarrow (\perp, \perp, \|\langle a_i \rangle - \langle b_i \rangle\|_H)$$

is private and uses just polylogarithmic communication. (Details about correctness and functional privacy are straightforward and omitted.)

Note that the view of each of Alice and Bob is a truly random string. The view of Carol is a pair of sketches derived from a pair of bit strings that are random except that they are at a prescribed Hamming distance. It follows that this protocol is a weakly private approximation protocol. On the other hand (as discussed below), it is not hard to see that Hamming-3 is not a private approximation protocol according to Definition 4. Definition 6 has several merits, but also has a number of problems:

• Definition 6 captures the intuition that the randomness in an approximation to a deterministic function f is like a nonce and not sensitive; only the value of the function is sensitive and should not be learned by

Hamming-3

1. Alice sends Bob a short random seed that they each expand to get a long shared pseudorandom string.
2. They use the shared pseudorandomness to agree on a permutation π of bit positions and a random n-bit mask, $\langle m_i \rangle$.
3. Using more of the shared pseudorandomness, Alice sketches $\langle a_{\pi(i)} + m_i \rangle$ and Bob sketches $\langle b_{\pi(i)} + m_i \rangle$. They send the short sketches to Carol. (They can use any of several appropriate sketching algorithms here.)
4. Carol approximately reconstructs $\|\langle a_{\pi(i)} + m_i \rangle - \langle b_{\pi(i)} + m_i \rangle\|_H = \|\langle a_i \rangle - \langle b_i \rangle\|_H$.

Fig. 3. A weakly private 3-party protocol for the Hamming distance $(\langle a_i \rangle, \langle b_i \rangle, \perp) \to (\perp, \perp, \|\langle a_i \rangle - \langle b_i \rangle\|_H)$

others. This is a reasonable assumption in some applications. In applications where Definition 6 is reasonable, it allows efficient protocols such as Hamming-3, which would be excluded by the stronger Definition 4.

• In traditional secure multiparty computation, each player's view must simulatable when concatenated with other players' outputs, but may be unsimulatable when concatenated with other players' views. This is because other honest players use their outputs but not their views in subsequent or parallel protocols. This allows the intuition that "no player learns anything not implied by her own input and output," even though, generally, each player learns about the others' views. Note that approximate output resembles traditional output in some respects and traditional view in other respects. In particular, it is natural for players to use approximate output in subsequent protocols, as, traditionally, they would use exact output. From this perspective, Definition 6 is unsatisfactory because some players can learn about the approximate outputs of other players, which might be used in future protocols or real life.

• Observe that the above protocol is insecure against a coalition of Alice and Carol. Together, Alice and Carol know the permutation, mask, sketching randomness, and a sketch of Bob's input. They can use this, for example, to form a sketch of Bob's original input, whence they learn an approximation to the Hamming weight of Bob's original input. We don't know of any two-party weakly private protocol for the Hamming distance that is more efficient than the protocol we give in Section 3.5, nor even a three-party weakly private protocol in which Alice or Bob learns the approximate output. Thus, when an input holder is to learn the approximate Hamming distance, the protocol of Section 3.5 is the best we have.

• For Hamming distance sketching protocols π of which we are aware, if the parties use π in performing protocol Hamming-3, Alice will learn something about the bias of Carol's output. To achieve semantic security, Alice should be able to simulate her view even if she knows that Bob's input is one of two possibilities. For example, suppose Alice holds the zero input and knows that Bob holds one of two inputs b_1 or b_2, of Hamming weight d. Suppose they use an ideal sketching algorithm that produces a uniformly random result in the range $d(1 \pm \epsilon)$. Since Alice knows the random permutation, mask and randomness used in the sketching algorithm, she knows $\hat{f}(0, b_1)$ and $\hat{f}(0, b_2)$. With probability $1/2$, either both $\hat{f}(0, b_1) > d$ and $\hat{f}(0, b_2) > d$ or both $\hat{f}(0, b_1) < d$ and $\hat{f}(0, b_2) < d$; *i.e.*, the approximation is, for all of Bob's inputs, too high or too low, and Alice knows which. A similar (though less dramatic) statement can be made when Bob's inputs are more general.

• It is problematic to modify Definition 6 for the secure case.

• A private approximation protocol according to Definition 4 also is a weakly private approximation protocol according to Definition 6. The simulators for players who do not get approximate output remain the same. For P_1, who does get approximate output, a simulator under Definition 6 for P_1's view can be constructed by composing the hypothesized simulators under Definition 4 for the approximate output from the exact output and for the view from the approximate output.

• Allowing blowup by polynomial factors, a function that has a weakly private approximation protocol according to Definition 6 also has a private approximation protocol according to Definition 4. This is because a weakly private approximation protocol outputs a functionally private approximation, which can be computed privately in polynomial time using standard techniques.

Because Definition 6 suffers some problems in the sublinear context and collapses to the more conservative Definition 4 modulo polynomial factor blowup, we present Definition 4 as our working definition, while reserving Definition 6 for specialized applications. Thus, even though protocol Hamming-3 is very efficient, we are forced at this point to abandon it as non-private in most situations.

A.2 Approximations that Yield Less Information Than Exact Computation

Our private approximation definition requires that $\hat{f}$ leaks no more about its inputs and outputs than f does. By definition, an approximation typically gives less information about the output of f than an exact computation of f would. For some applications, one might also want to require that the approximation function reveals strictly less than the target function: for example, that $\hat{f}(\mathbf{x})$ does not reveal $f(\mathbf{x})$. However, it appears that the formal requirement needed would depend on the specific approximation requirement and functions being considered.

We now briefly (and speculatively) consider two different ways of limiting the amount of information learned about f from $\hat{f}$.

Imprecision: If the output values z_1 and z_2 are close, then the distributions on their approximations, $\hat{z}_1$ and $\hat{z}_2$, are nearly indistinguishable. (The cost to distinguish depends on the closeness of z_1 and z_2.)
Repeatability: Two independent samples from $\hat{f}$ yield no more information than a single sample from $\hat{f}$.

Each of the above security requirements can be achieved by standard approximation techniques. Suppose f is a real-valued function that takes on all possible outputs in a large range. For a fixed ϵ, first suppose that $\hat{f}$ is a Gaussian or uniform distribution centered at f with width ϵ. Then $\hat{f}$ approximates f (under the appropriate approximation requirement), and satisfies imprecision but not repeatability. Now suppose $\hat{f}$ is f rounded to the nearest power of $(1+\epsilon)^2$; i.e., the nearest even power of $(1+\epsilon)$. Then $\hat{f}$ approximates f (under the appropriate approximation requirement), and satisfies repeatability but not imprecision.

There is no way, however, to achieve simultaneously both imprecision and repeatability. If the approximation is randomized, then repeated independent samples will yield more information (at least in the statistical and perfect senses). On the other hand, if the approximation is deterministic and non-constant then there must be two close values, z_1 and z_2, with different approximations (and therefore z_1 and z_2 are distinguishable by their approximations). Thus there is no definition of secure approximation that forces both imprecision and repeatability without restricting the approximation requirements that can be supported.

Furthermore, it is desirable to be able to support both of the kinds of approximation described above. Neither alone gives complete information about f, and, as we now argue, both are important from computational and cryptographic standpoints. First, we consider the computational aspects. An approximation of a value v by a Gaussian centered at v arises often in (the modeling of) physical measurements. Also, the even power of $(1+\epsilon)$ nearest to v is similar to a floating point representation for v; these representations are useful in computation. Now, we consider the cryptographic aspects. First assume that we are given an approximation f' to f with a guarantee that $|f'-f| < \epsilon 2^{-k}$, but such that $f'-f$ contains information about an input value. As is discussed in Appendix A.3, by outputting $\hat{f} = f'(1+X) \stackrel{c}{\equiv} f(1+X)$ (these are statistically indistinguishable, where X is a uniform distribution centered at zero with width ϵ), we construct a private approximation $\hat{f}$ to f. Next, suppose we are given an approximation f' to f and that both are guaranteed to be far from odd powers of $(1+\epsilon)$. By defining $\hat{f}$ to be f' rounded to the nearest even power of $(1+\epsilon)$ (which equals f so rounded, by assumption about the goodness of the approximation f'), $\hat{f}$ is a perfectly private approximation to f.

It is possible to enforce either imprecision or repeatability via simulators and indistinguishability of distributions. Since some applications require imprecision while others require repeatability, and since it is not possible to achieve both simultaneously, an application should add the appropriate security requirement to our Definition 1, as needed.

A.3 Random Noise and Rounding

In this section, we note that the obvious approach of taking an insecure approximation and making it secure by adding in random noise or masking the low-order bits does not generally work. There are, however, some cases in which it can be useful. We first show that rounding does not generally provide functional privacy.

Next, we show that adding random noise does provide functional privacy, but is not generally efficient. We then show that in the case of finite-precision approximations to real-valued functions, adding random noise does provide an efficient solution.

Rounding. Consider taking an approximation $\hat{f}$ for f which is good to within $(1 \pm \epsilon/3)$ with high probability, and rounding it down to a power of $(1+\epsilon/3)$. One can check that the result is in the range $(1 \pm \epsilon)f$ with high probability. But, now consider a function f whose approximation takes on all real values within a large range, with high precision, as both the inputs and the source of randomness vary. Suppose there are two sets of inputs to f, $\mathbf{x}$ and $\mathbf{x}'$, such that $f(\mathbf{x}) = f(\mathbf{x}')$, $x_1 = x'_1$, but $\hat{f}(\mathbf{x})$ and $\hat{f}(\mathbf{x}')$ are distributed differently (if $\hat{f}(\mathbf{x}) \stackrel{c}{\equiv} \hat{f}(\mathbf{x}')$, then $\hat{f}$ already satisfies our definition of functional privacy with respect to f, even before rounding, so we need not consider this). That is, for one or more t, $\Pr(\hat{f}(\mathbf{x}) < t) \neq \Pr(\hat{f}(\mathbf{x}') < t)$. Since $(x_1, f(\mathbf{x})) = (x'_1, f(\mathbf{x}'))$, we require that the private approximation distributions $\hat{f}(\mathbf{x})$ and $\hat{f}(\mathbf{x}')$ be the same for $\mathbf{x}$ and $\mathbf{x}'$. But, if we are unlucky in the value(s) of t, which is likely to happen if f and $\hat{f}$ take on all values in a large range, then $\Pr\left((1+\epsilon/3)^i \leq \hat{f}(\mathbf{x}) < (1+\epsilon/3)^{i+1}\right) \neq \Pr\left((1+\epsilon/3)^i \leq \hat{f}(\mathbf{x}') < (1+\epsilon/3)^{i+1}\right)$. It follows that $\hat{f}$ rounded down to a power of $(1+\epsilon/3)$ is not functionally private with respect to f. Nevertheless, if $\hat{f}$ (miraculously) also always satisfies $(1+\epsilon/3)^i \leq \hat{f} \leq (1+\epsilon/3)^{i+1/2}$, then the rounding technique will work, as can be checked readily. Somewhat weaker conditions are also possible, but in general, rounding will not provide functional privacy.

Adding random noise. Suppose we are given an approximation scheme for f, i.e., for any $\epsilon, \delta > 0$, we can output a number that is within the factor $(1 \pm \epsilon)$ of f with probability $1 - \delta$. We can then construct a secure approximation as follows. Given security parameter k such that two distributions are considered statistically indistinguishable if their statistical difference is no more than 2^{-k}, first construct an approximation z' to an output z of f that is good to within the factor $(1 \pm 2^{-k}\epsilon/2)$. Next, let $\hat{z} = z'(1+X)$, where X is uniformly random on the interval $[-\epsilon/2, \epsilon/2]$. One can readily check that this procedure yields an approximation scheme for f and that the final output, $\hat{z}$, is a private approximation to z.

Unfortunately, this procedure is not efficient unless the approximation z' is so good as to be usable to obtain an essentially exact solution, or unless k is very small. By definition, if f is hard to compute then an approximation good to within the factor $(1 \pm \epsilon)$ requires time more than polylog in $1/\epsilon$ to compute, so the above procedure requires time more than polynomial in k. Nevertheless, if k can be taken small enough, this procedure is a simple and straightforward solution.

Finite-Precision Approximations to Real-Valued Functions. Suppose Alice and Bob wish to compute a discrete-valued or real-valued function whose specification depends on real-valued functions like the logarithm or square root. In practice, some finite precision approximation will be used for intermediate values as well as the final output, if the latter is, ideally, real-valued instead of discrete. For example, Lindell and Pinkas [LP00] give a privacy-preserving protocol to construct a (discrete) decision tree from data shared between Alice and Bob. The tree is constructed by starting with a single node and iteratively deciding to replace a leaf with a binary tree of height one. The leaf to split is the one with maximum *entropy gain*. That is, one needs to find the maximum, over all leaves, of a quantity like $-x_\ell \ln(x_\ell)$ where x_ℓ is some arbitrary rational number associated with leaf ℓ. They use a k-term Taylor expansion for the logarithm, *i.e.*, a deterministic approximation good to within $\left(1 \pm 2^{-k}\right)$. In general (especially using randomized approximations), one might worry that $x \ln x < y \ln y$ but $x\tilde{\ln} x > y\tilde{\ln} y$, where $\tilde{\ln}$ denotes the approximation to the logarithm. This would result in a different discrete output. Typically, the risk to correctness is minor—all possible outputs will be good enough—but the risk to privacy is important, since a single changed bit in a discrete output can mean the difference in leaking information about private inputs.

Fortunately, there is a standard procedure to avoid these difficulties. Alice and Bob use an approximation to within $\left(1 \pm 2^{-2k}\right)$, then multiply $(1 \pm X)$, where X is uniformly random in $[-2^{-k}, 2^{-k}]$. As noted above, the result will be functionally private. In typical computations involving elementary or algebraic functions, correctness relies only on the existence of some $\left(1 \pm 2^{-k}\right)$ approximation, not on any particular approximation (otherwise, a portable implementation would be difficult for a variety of a finite-precision computers, even without worrying about privacy). For such computations, the technique of adding random noise will result in a correct output. In the case of building a decision tree, for example, the ideal specification is deterministic and requires a single output from a given input. An implementation of the algorithm will produce one of

several trees, depending on the arbitrary way finite precision is handled. The noise-added algorithm will produce a decision tree within the range of those considered to be correct on limited-precision machines, but from a distribution that doesn't depend on the input.

B Additional protocols

B.1 A protocol based on Reed-Solomon codes

In this section, we describe a private sketching protocol for the Hamming distance that is correct (without any probability of error) provided the Hamming distance is at most $d_{\max}$. This can be used in Section 3.4.

We start by describing a basic version of this protocol which, while in principle is sufficient for our needs, has some disadvantages (such as its round complexity). We then proceed to describe the improved protocol, achieved by modifying the basic version. We note that in the basic version the function g will be functionally private only when the distance is low, but in the improved version it is private for any distance (satisfying the properties discussed in the beginning of Section 3.4).

Basic Version. Let F be a finite field, where $|F| > n$. We view the inputs a, b as vectors in F^n. Let H be the parity-check matrix of a Reed-Solomon code over F with distance $2d_{\max}+1$, dimension n, and length $n+2d_{\max}$. The matrix H has $2d_{\max}$ rows and n columns. For any $x \in F^n$ such that $w_h(x) \leq d_{\max}$, x can be uniquely recovered from the *syndrome* Hx (since x can be viewed as a corrupted encoding of 0). The above facts imply the following (non-private) sketching protocol for the Hamming distance, *given the promise that it is smaller than* $d_{\max}$. The sketching function is deterministic and is defined by $S(x) = Hx$. Reconstruction proceeds as follows. From the syndromes Ha and Hb, one can compute the syndrome $H(a-b)$. The output is computed by recovering $a-b$ from its syndrome and outputting its weight. By choosing a field F of size $O(n)$, the sketch size is $O(d_{\max} \log n)$. As follows from the known methods for decoding Reed-Solomon codes, the circuit complexity of the reconstruction function is $\tilde{O}(d_{\max})$. Thus, using generic private circuit evaluation [Yao86,GMW87], $d = d_h(a,b)$ can be privately computed with $\tilde{O}(k \cdot d_{\max})$ communication, given the promise that $d \leq d_{\max}$.

The output function g induced by the above sketching protocol fails to be functionally private when the distance is larger than $d_{\max}$. This follows from the fact that there exist x, x' such that $w_h(x) = w_h(x') > d_{\max}$ and yet applying the decoding procedure to Hx and Hx' yields a different number of errors. In particular, the above private protocol will fail to be private when the distance is large. In contrast, the privacy of our protocol for the high distance case holds for *any* choice of inputs, regardless of the distance d. This state of affairs already allows us to obtain a final private protocol with the following "cautious" two-stage structure. First, the private estimator for the high distance case is invoked. Then, only if its output provides overwhelming evidence that the distance is too low to be reliably approximated, the low distance protocol is invoked.

The above two-stage protocol suffers two disadvantages. First, it fails to be private in a scenario where only one of the two parties is supposed to learn the output (since the second party will learn partial information on the output from the communication pattern). Second, it does not achieve the best possible round complexity. This motivates the following modification of the low-distance sketching protocol, which guarantees that the output g is *almost determined* by d_h over the entire range of inputs (see discussion in the beginning of Section 3.4). Moreover, the circuit size of the reconstruction function will not be significantly increased.

Improved Protocol. The sketching function of the modified sketching protocol will use a k-bit random input r, where r is interpreted as a key to a pseudo-random function $h_r : [n] \to \mathrm{GF}(2)^k$. The n possible outputs of h_r define a pseudo-random $k \times n$ matrix R over GF(2), satisfying the following properties: (1) the i-th column of R can be computed from r by a circuit of size $\tilde{O}(k)$; (2) for any nonzero $x \in \mathrm{GF}(2)^n$, the probability that $Rx = 0$ is negligible in k, where the probability is over the uniform choice of r from $\{0,1\}^k$. (We use *general* pseudo-random functions for simplicity; more efficient constructions can be based on small-bias probability spaces [NN90].) The sketching function is defined by $S(x,r) = (Hx, Rx, r)$, where R is the $k \times n$ matrix defined by h_r. Reconstruction proceeds as follows. First, Ha and Hb are used as before to "decode" $H(a-b)$. However, instead of only counting the *number* of errors, this time we will also use their *locations* to test reliably whether a, b differ exactly in the specified places. Let v_e denote the error vector produced by the decoding algorithm from $H(a-b)$. Note that $w_h(v_e) \leq d_{\max}$, and that $v_e = (a-b)$ if and only if $d_h(a,b) \leq d_{\max}$. The reconstruction procedure tests whether $Ra - Rb - Rv_e = 0$. If the test

succeeds, the reconstruction function outputs the *number* of errors, and otherwise it outputs "fail". From the above properties of h_r we may conclude: (1) reconstruction can be implemented by a circuit of size $\tilde{O}(k^{O(1)} \cdot d_{\max})$; (2) if $d = d_h(a,b) \leq d_{\max}$, $g(a,b) = d$ with probability 1; (3) if $d > d_{\max}$, then $g(a,b)$ outputs "fail" with overwhelming probability. Our final sketching protocol thus satisfies all the desired properties, and can therefore be computed privately and efficiently as explained in the beginning of Section 3.4.

B.2 Protocols with Offline Communication

In this section we obtain efficient private approximation protocols for the following scenario. Suppose that Alice and Bob are allowed to communicate $\tilde{O}(n)$ bits at zero cost before they receive their inputs. We charge them only for *online* communication performed *after* they learn their inputs. In this model, we give private protocols with only $\tilde{O}(1)$ communication cost.

We consider the L^2 distance $\left(\sum |a_i - b_i|^2\right)^{1/2}$, where $\langle a_i \rangle$ and $\langle b_i \rangle$ are sequences of integers.[6] A solution for the Hamming distance follows as a special case. Essentially, we verify that the protocol from [Ind00] is functionally private and can be efficiently implemented by a private protocol in this model.

Alice and Bob share a vector $\langle s_i \rangle$ of n samples from a Gaussian distribution.[7] These samples are encrypted using *homomorphic public-key encryption*, *i.e.*, anyone can form an encryption $E(\alpha, \kappa)$ of α that can be decrypted only by knowing the secret key, κ, and, from encryptions $E(\alpha, \kappa)$ and $E(\beta, \kappa)$ of α and β for the same secret key κ, anyone can form an encryption $E(\alpha + \beta, \kappa)$ of $\alpha + \beta$ for κ. By using a *threshold* homomorphic encryption scheme, Alice and Bob split κ so that neither can decrypt alone but together they can decrypt.

As prescribed in [Ind00], Alice should form $\sum_i a_i s_i$. In our context, she forms $E\left(\sum_i a_i s_i, \kappa\right)$, as follows. She forms $E(a_i s_i, \kappa)$ from $E(s_i, \kappa)$ and a_i, in time $k^{O(1)} \log a_i$, using the homomorphic properties of the encryption and repeated doubling. She then forms $E\left(\sum_i a_i s_i, \kappa\right)$, using the homomorphic properties of the encryption. Alice and Bob then form $E\left(\sum_i s_i (a_i - b_i), \kappa\right)$, again using the homomorphic properties of the encryption. The insecure protocol prescribes that they compute $\left(\sum_i s_i (a_i - b_i)\right)^2$, repeat, and take medians of means, using Lemma 5 (proven in Appendix C.1). In our setting, Alice and Bob perform the median of means of squares of decryptions of $E\left(\sum_i s_i (a_i - b_i), \kappa\right)$-values using a secure multiparty computation (this is a small circuit). Correctness is easy to verify, using the fact that the expected value of $s_i s_j$ is 1 if $i = j$ and 0 otherwise (or see [Ind00]). Privacy of the messages is immediate by construction.

As for functional privacy, first observe that the result depends on $\langle (a-b)_i \rangle$, but not otherwise on $\langle a_i \rangle$ or on $\langle b_i \rangle$. Also, Alice and Bob are allowed to learn $\|\langle a_i \rangle - \langle b_i \rangle\|_2$, that is, the Euclidean distance between their inputs. It is a well-known property of the Gaussian distribution that the product $\langle s_i \rangle$ of Gaussians is a spherically-symmetrical distribution. Functional privacy follows immediately.

C Proofs

C.1 A Folklore Lemma

The following lemma is used in several of our proofs.

Lemma 5. *Let X be a real-valued random variable such that, for some c, $E[X^2] \leq c \cdot \mathrm{var}[X]$. Then, for any $\epsilon, \delta > 0$, there exists a random variable Z such that $\Pr(|Z - E[X]| \geq \epsilon E[X]) \leq \delta$, and Z is a function of $O(c \cdot \log(1/\delta)/\epsilon^2)$ independent samples of X.*

Proof: Let Y be the average of $8c/\epsilon^2$ independent copies of X. Then $E[Y] = E[X]$ and $\mathrm{var}[Y] \leq \epsilon^2 E^2[X]/8$. By the Chebychev inequality, $\Pr(|Y - E[X]| > \epsilon E[X]) \leq \frac{\mathrm{var}(Y)}{\epsilon^2 E^2[X]} \leq \frac{1}{8}$. Let Z be the median of $4\log(1/\delta)$ independent copies of Y. Then $|Z - E[X]| \geq \epsilon E[X]$ iff for at least half the Y_i's, $|Y_i - E[X]| \geq \epsilon E[X]$. Since, for each i, this happens only with probability $1/8$, the Chernoff inequality implies that $\Pr(|Z - E[X]| \geq \epsilon E[X]) \leq \delta$. ∎

[6] The square of the L^2 distance, $\sum |a_i - b_i|^2$, is equivalent to the L^2 distance from the perspective of computation and privacy. Henceforth, we consider the easier-to-read square of the L^2 distance.

[7] Actually, the finite-precision samples will be statistically indistinguishable from Gaussians, which is good enough.

C.2 Proof of Lemma 1

Lemma 1 Private-Sample-XOR *is a private protocol computing the randomized function* Sample-XOR.

Proof (sketch): The correctness of the protocol follows by observing that $z_A = (b \ll r_B)_{r_A} \oplus m_B = b_{(r_A+r_B)} \oplus m_B$ and, symmetrically, $z_B = a_{(r_A+r_B)} \oplus m_A$ (where addition of indices is taken modulo n). Hence, both players output

$$\begin{aligned} z'_A \oplus z'_B &= z_A \oplus z_B \oplus m_A \oplus m_B \\ &= a_{(r_A+r_B)} \oplus b_{(r_A+r_B)} \end{aligned}$$

where $r = r_A + r_B$ is a uniformly distributed index.

The privacy of the protocol follows from the fact that in the process of obtaining the output $a_{(r_A+r_B)} \oplus b_{(r_A+r_B)}$, no party learns $r_A + r_B$, $a_{(r_A+r_B)}$, or $b_{(r_A+r_B)}$. A simulator for Alice's view may proceed as follows. On input $a \in \{0,1\}^n$ and output value $z \in \{0,1\}$:

1. Pick at random z'_A, z'_B such that $z'_A \oplus z'_B = z$, $r_A \stackrel{\mathrm{R}}{\leftarrow} [n]$, and $m_A \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}$.
2. Let $z_A \stackrel{\mathrm{def}}{=} z'_A \oplus m_A$. Invoke the simulator of the $\binom{n}{1}$-OT protocol twice, once with Alice as a receiver having input r_A and output z_A, and once with Alice as a sender having input $(a \ll r_A) \oplus m_A$. Let $view_1, view_2$ denote the views produced by the two simulations.
3. Output $(a, r_A, m_A, view_1, view_2, z'_B)$.

A simulator for Bob's view may be obtained similarly. ∎

C.3 Proof of Theorem 2

In this section, we provide a more detailed proof of Theorem 2, namely that our modification of the [JSV00] result is a private approximation to the permanent.

We will use the standard polynomial-time private computation techniques to insure that the intermediate messages are private. Thus it suffices to show how to approximate the permanent in a functionally private way. To this end, we use ideas of [JSV00].

The of technique [JSV00] uses a rapidly-mixing Markov chain to show how to sample from the set of all perfect matchings on a graph from a distribution that is statistically indistinguishable from uniform. This can be used to approximate the permanent as follows. Let p_e denote the probability that edge e is in a random matching, and let $p_{e|E'}$ denote the probability that e is in a random matching given that edges in $E' \subseteq E$ are in the matching. Below we will find a sequence $e_1, e_2, \ldots, e_n$ of edges such that $p_{e_1}, p_{e_2|e_1}, p_{e_3|e_1,e_2}, \ldots$ are all at least $1/n$, so that, in particular, $\{e_1, \ldots, e_n\}$ is a perfect matching. Write the number of perfect matchings as

$$1/(p_{e_1} p_{e_1|e_2} \cdots p_{e_n|e_1,\ldots,e_{n-1}}).$$

To approximate p_{e_1}, sample (nearly) uniformly from all perfect matchings in the given graph G and count the fraction of times in which e_1 occurs. Since $p_{e_1} \geq 1/n$, we can sample $O\left(n^3/\epsilon^2\right)$ times and get a good relative error approximation to p_{e_1}, namely, $p_{e_1}(1 \pm O(\epsilon/n))$. This yields an approximation of the denominator, and, hence, the fraction, that is good to within the factor $(1 \pm O(\epsilon))$. To approximate $p_{e_2|e_1}$, consider the graph $H = G \setminus \{v_1, v_2\}$, where $e_1 = \{v_1, v_2\}$. Perfect matchings in H are in bijection with perfect matchings in G in which e_1 occurs. Thus we can approximate $p_{e_2|e_1}$ by sampling perfect matchings in H and counting the fraction of times that e_2 occurs.

To find a suitable sequence of edges, proceed as follows. There are at most n^2 edges and each matching has at least n edges, so some edge is in at least $1/n$ of the matchings. We can find such an edge, e_1, by sampling using the Markov chain. Similarly, there are at most $(n-1)^2$ edges remaining after the removal of e_1's endpoints, and each matching of the remaining graph has $n-1$ edges, so some edge, e_2, is in at least $1/(n-1)$ of those matchings. Again, we can find it by sampling. Continuing in this way generates the sequence of edges.

It is immediate that a sampling-based approximation to p_{e_1} depends only on p_{e_1}, and not on the rest of the graph. The Monte Carlo sampling-based approximation is statistically indistinguishable from depending

only on p_{e_1}. As an aside, we can conclude that the probability that a given edge is in a perfect matching in a given graph is privately approximable with good relative error, provided the probability is large enough (at least a negative power of n), and is privately approximable with good additive error in any case. From the discussion above, it is not difficult to see that the exact version of this promise problem is reducible from any of the $p_{e|E'}$'s, so the exact version is reducible from the permanent and is #P-hard.

But now consider two different graphs, G_1 and G_2, with the same value for the permanent but for which the corresponding values of p_{e_1}, $p_{e_2|e_1}$, etc., are very different. In general, the product of approximations $\hat{p}_{e_1}\hat{p}_{e_1|e_2}\cdots\hat{p}_{e_n|e_1,\ldots,e_{n-1}}$ for graph G may have a very different distribution than the product of approximations $\hat{p'}_{e_1}\hat{p'}_{e_1|e_2}\cdots\hat{p'}_{e_n}|e_1,\ldots,e_{n-1}$ for graph G', even though

$$p_{e_1}p_{e_1|e_2}\cdots p_{e_n|e_1,\ldots,e_{n-1}} = p'_{e_1}p'_{e_1|e_2}\cdots p'_{e_n|e_1,\ldots,e_{n-1}}.$$

As an illustration, suppose $p_{e_1} = p_{e_2|e_1} = 1/2$ but $p'_{e_1} = 1$ and $p'_{e_2|e_1} = 1/4$, so $p_{e_1}p_{e_2|e_1} = p'_{e_1}p'_{e_2|e_1} = 1/4$. If t samples are used to approximate each factor, then one over the number of perfect matchings is approximated as $B(.5,t)\cdot B(.5,t)/t^2$ for graph G and $B(.25,t)/t$ for graph G', where $B(p,t)$ is the number of successes in t independent trials of an experiment with success probability p. We show that the logarithms of these two estimators, L_1 and L_2, respectively, are distinguishable, whence the estimators themselves are distinguishable. The former quantity L_1 has distribution

$$\begin{aligned}\log(1/4 \pm \Theta(1/\sqrt{t})) &\approx \log(1/4) \pm \log(1+\Theta(1/\sqrt{t}))\\ &\approx \log(1/4) \pm \Theta(1/\sqrt{t})\\ &\approx \log(1/4) \pm X_1,\end{aligned}$$

whereas, similarly, L_2 has distribution roughly

$$\log(1/4) \pm X_2 \pm X_3,$$

where X_1, X_2, and X_3 are independent and uniformly distributed on $[-\Theta(1/\sqrt{t}), \Theta(1/\sqrt{t})]$. The density function of L_1 is a single pulse of width $\Theta(1/\sqrt{t})$, whereas the density function of L_2 is the convolution of two such pulses, *i.e.*, an isosceles triangle of width $\Theta(1/\sqrt{t})$, symmetric about $\log(1/4)$. These distributions have the same mean, but they are statistically distinguishable unless t is exponential in k. Note that this leakage of information would occur even if one could sample perfectly from the set of all perfect matchings; this leakage has nothing to do with the statistical difference between the Markov-Chain-based sample and a truly uniform sample.

To avoid this leakage, one might be tempted to estimate the product at once; that is, sample from the joint distribution of perfect matchings of G, perfect matchings of $G \setminus$ vertices of e_1, etc., and estimate the probability of the joint event that e_1 occurs in a matching of G and e_2 occurs in a matching of $G\setminus$vertices of e_1, etc. But, after simplifying, this amounts to estimating the probability that a random perfect matching of G is the one matching $\{e_1,\ldots,e_n\}$. In general this probability is minuscule, even if each $p_{e|E'}$ has large probability, so the estimate will be zero, which gives no estimate for the size of the permanent.

To remedy this, we will manipulate success probabilities in various ways. One important process we call *enriching*. Suppose we are given a coin with success probability p, whose value is unknown except that $p \ll 1/t$ for some real number $t \geq 1$. We *enrich* the coin by the factor t when we do the following experiment: Toss the original coin a large number N of times so that we can ignore the possibility that the number S of heads is greater than N/t. Now toss a coin with success probability tS/N. We claim that this latter coin has success probability exactly tp.

To see this, observe that if $t = 1$ then the statement is true, since the constructed coin can be viewed as a random trial from among the N tosses of the original coin. (In fact, an arbitrary trial of the original coin would have the right probability, p.) Write the probability that the constructed coin succeeds as $p = \sum_s \Pr(1|S=s)\Pr(S=s) = \sum_s \frac{s}{N}\Pr(S=s)$. On the other hand, for general t, the probability of success of the constructed coin is

$$\begin{aligned}\sum_s \frac{ts}{N}\Pr(S=s) &= t\sum_s \frac{s}{N}\Pr(S=s)\\ &= tp.\end{aligned}$$

Note that enriching is only possible if the original probability, p, is at least $1/n^{O(1)}$. This prevents us from directly enriching the probability $p_{e_1}p_{e_2|e_1}\cdots p_{e_n|e_1,\ldots,e_{n-1}}$. Also, enriching increases the number of original coin tosses by a factor of the security parameter, k. This prevents us from enriching recursively to more than constant depth, so we cannot enrich $p_{e_1}p_{e_2|e_1}\cdots p_{e_n|e_1,\ldots,e_{n-1}}$ by writing this product as a tree of bounded-fanin products and enriching the output of each such bounded-fanin product.

Thus, besides enriching, we will need to do other constructions on success probabilities. Given coins with success probabilities p, q and r, we can form a coin with success probability pq by taking the joint event, probability $1-p$ by taking the complementary event, probability t, for known $t \leq 1$, by constructing from scratch, and probability $rp + (1-r)q$, by flipping r and then either flipping p or q.

Before proceeding, we note that if k and n are more than exponentially far apart, there's a trivial algorithm. First, if the security parameter k is tiny, so that $2^k < n$, then we can use the noise-adding technique of Section A.3. In time polynomial in n and ϵ, estimate each factor to within $\left(1 \pm O\left(\epsilon/\left(n2^k\right)\right)\right)$. This estimates the product to within $\left(1 \pm O\left(\epsilon/2^k\right)\right)$. Finally, multiply by some noise in the range $(1 \pm O(\epsilon))$. The result will be good to the within the factor $(1 \pm O(\epsilon))$ and, for any two matrices with the same permanent, the statistical difference between the output distributions will be at most 2^{-k}. On the other hand, if k is huge, so that $2^n < k$, then, as shown by Ryser in [Minc82], we can solve the permanent problem exactly in time polynomial in k.

We return now to permanents. Sample each probability $p_{e_i|e_1,\ldots,e_{i-1}}$ to determine its rough value, then scale it up to $\Omega(1)$ by enrichment. This uses a factor $O(k)$ more samples from each distribution of matchings. Keep track of the chosen scaling factor, ϕ. For x at least $\Omega(1)$, a $O(k)$-term Taylor expansion for $x^{1/n}$ around $x = 1$, *i.e.*, $\sum(-1)^j\binom{1/n}{j}(1-x)^j$, will have error bounded by $2^{-2k} \leq 2^{-k}/n$. (Recall we are assuming that $n < 2^k$.) The coefficient of $(1-x)^j$ is

$$\begin{aligned}(-1)^j\binom{1/n}{j} &= (-1)^j\frac{\left(\frac{1}{n}\right)\left(\frac{1}{n}-1\right)\left(\frac{1}{n}-2\right)\cdots\left(\frac{1}{n}-j+1\right)}{j!}\\ &= -\frac{\left(\frac{1}{n}\right)\left(1-\frac{1}{n}\right)\left(2-\frac{1}{n}\right)\cdots\left(j-1-\frac{1}{n}\right)}{j!}\\ &= -\frac{1}{nj}\left(1-\frac{1}{n}\right)\left(1-\frac{1}{2n}\right)\cdots\left(1-\frac{1}{(j-1)n}\right),\end{aligned}$$

i.e., at most $1/(nj)$ in absolute value, and negative. Thus the sum of the absolute values of all but the leading coefficient in a $O(k)$-term Taylor series is at most $\sum_{j=1}^{k}\frac{1}{nj} \leq \frac{\log k}{n}$, which we can assume is less than 1, since, otherwise, if $k > 2^n$, we can solve the permanent exactly in time polynomial in k. That is, the $O(k)$-term Taylor series is 1 less a subconvex combination of $x, x^2, \ldots, x^k$. Thus, given a coin with unknown success probability p at least $\Omega(1)$, we can construct, using the $p, q \to pq$, $p \to 1-p$, $p \to tp + (1-t)q$ and t constructions, an experiment whose success probability is $p^{1/n}(1 \pm 2^{-k}/n)$, using at most $O(k)$ original samples, enough to evaluate a polynomial of degree k. Enriching p from $\Theta(1/n)$ to $\Omega(1)$ requires an additional factor of k samples. Finally, we can divide out by $\phi^{1/n} \approx 1$ by using the $p \to tp$ construction. This way we'll have constructed $p'_{e|.} = p_{e|.}^{1/n}$ for each probability $p_{e|.}$.

As an illustration, consider the three-term expansion to the square root of x at $x = 1$, namely

$$\sqrt{x} \approx T(x) = 1 - \frac{(1-x)}{2} - \frac{(1-x)^2}{8}.$$

Suppose event A has probability p and suppose F_t (a coin flip) has success probability t. Rewrite $T(x)$ using Horner's rule to decrease the number of multiplications and using convex combinations instead of sums, getting

$$T(x) = 1 - \left[(1-x)\frac{1 + (1-x)/4}{2}\right].$$

Let $(A\ ?\ B : C)$ denote the experiment of performing A, if it is successful then performing and outputting the result of B, otherwise performing and outputting the result of C. The following event, which can be

constructed directly from the above form for $T(x)$, has success probability $T(p)$, where each occurrence of A and F is independent.

$$\overline{\overline{A}(F_{1/2} \,?\, F_1 : \overline{A}F_{1/4})}. \tag{2}$$

For the polynomial of degree two, Experiment (2) uses just two A experiments and constantly-many other experiments for each A experiment, though, in general, constructing F_t from $F_{1/2}$ may require $\Omega(k)$ repetitions to achieve the desired accuracy. The coefficients other than the leading 1 sum to less than 1, so the sum of this part of the series can be implemented using the $(\cdot \,?\, \cdot : \cdot)$ construction and the construction of taking a joint event with some F_t.

Ideally, the joint distribution $p'_{e_1}, p'_{e_2|e_1}, \cdots$ will have success probability $M^{-1/n}$, where M is the permanent. If each factor has success probability correct to within $\left(1 \pm 2^{-k}/n\right)$, then the joint event will have success probability correct to within $\left(1 \pm 2^{-k}\right)$. That is, for any two graphs with the same number M of perfect matchings and whatever arbitrary choices we make in scaling probabilities to $\Omega(1)$, we will construct an experiment with success probability statistically indistinguishable from $M^{-1/n}$. This analysis is for privacy—a bad guy making exponentially-many samples won't be able to distinguish the success probability from $M^{-1/n}$. With regard to correctness, we won't be able to recover all that accuracy using only a polynomial number of samples, but we will be able to recover sufficient accuracy. Note that, since each probability $p_{e|\cdot}$ is between $1/n$ and 1, so is their geometric mean, $M^{-1/n}$. Thus the average of n coin tosses will have the same mean, $M^{-1/n}$, and variance at most $M^{-1/n}/n$, which is at most $O\left(\left(M^{-1/n}\right)^2\right)$. We can then apply Lemma 5 with distortion ϵ/n and error probability δ, getting $M^{-1/n}(1 \pm O(\epsilon/n))$ with probability at least $1 - \delta$. It follows that the $-n$ power is $M(1 \pm O(\epsilon))$.

C.4 General Techniques Based on Monte Carlo Methods

In Section 4, we gave a private approximation protocol for the permanent of a matrix shared by two players. In this appendix, we generalize those techniques.

We consider intractable functions $f(a, b)$ that have polynomial-time approximation schemes. In this context, we seek a private approximation protocol with polynomially-bounded communication and computation, but we do not otherwise constrain the communication or computation. Therefore, we need only provide a functionally private approximation $\hat{f}$ to f; we can then draw upon the polynomial-time secure multiparty computation literature to compute $\hat{f}$ privately or securely.

In this section we consider only $\langle \epsilon, \delta \rangle$-approximations. Other approximation requirements can be considered, too.

In the following, we assume that there is an underlying size n and security parameter k. Computations must be correct to within the factor $(1 \pm \epsilon)$ with probability 3/4. Two distributions are "statistically indistinguishable" if their statistical difference is at most 2^{-k} (and a condition of similar strength in k applies for computational indistinguishability). "Polynomial time" means time polynomial in n, k, and $1/\epsilon$, and is denoted here by "poly." As usual, the failure probability 3/4 can be boosted up to $1 - \delta$ by performing $O(\log(1/\delta))$ repetitions.

We begin with a definition that says ψ is an approximation-preserving function.

Definition 7. *A deterministic real function ψ is* polynomially relatively continuous *if, for all x and for all $\epsilon > 0$, there exists $\eta > 1/\operatorname{poly}$ such that $\psi(x(1 \pm \eta)) \subseteq \psi(x)(1 \pm \epsilon)$.*

Lemma 6. *Let ψ be a polynomially relatively continuous function that is easy to compute and to invert. Suppose $f(a, b) = \psi(\Pr(\mathcal{E}))$, $\Pr(\mathcal{E}) \geq 1/\operatorname{poly}$, where $\mathcal{E}$ is an event (parameterized by a and b) under a probability distribution, D, such that, in polynomial time, one can sample from a distribution that is computationally indistinguishable from D. Then $f(a, b)$ has a functionally private approximation computable in polynomial time.*

Proof. One can estimate $\Pr(\mathcal{E})$ to within the factor $(1 \pm \eta)$ in polynomial time using Lemma 5, then apply ψ. To see that this is functionally private, note that from $f(a, b)$ alone (even without an additional input a or b), a simulator can construct an $\Omega(k)$-bit approximation to $\Pr(\mathcal{E}) = \psi^{-1}(f(a, b))$. It can then sample from a distribution with success probability indistinguishable from $\Pr(\mathcal{E})$, and apply ψ. The result follows.

In general, as in the case of the permanent, a Monte Carlo Markov chain approach to approximations involves making several estimates from separate Markov chain experiments and combining the estimates in an arbitrary way. While we cannot claim that any function with a Monte Carlo Markov chain-based approximation also has a functionally private approximation, we do exhibit functionally private approximations for a large class of such functions.

Lemma 7. *Let ψ be a polynomially relatively continuous function that is easy to compute and to invert. Suppose $f(a,b) = \psi(\phi(\Pr(\mathcal{E}_1), \Pr(\mathcal{E}_2), \ldots \Pr(\mathcal{E}_j)))$, where each event has probability at least $1/\operatorname{poly}$ in a probability distribution that can be nearly sampled in polynomial time, and where ϕ is a polynomial-sized, constant-depth arithmetic formula with gates of the following form:*

- $t \to 1 - t$
- $t_1, t_2 \to t_1 t_2$
- $\perp \to r$, *where* $r \in [1/\operatorname{poly}, 1 - 1/\operatorname{poly}]$
- $(t_1, t_2, \ldots, t_\ell) \to \sum_i r_i t_i$, *where* $\sum_i r_i = 1$
- $t \to t^r$, *for* $1/\operatorname{poly} \leq r \leq 1$
- $t \to rt$, *for* $r \geq 1$, *under the promise that* $rt < 1 - 1/\operatorname{poly}$.
- $(t_1, t_2, \ldots, t_\ell) \to \prod_i t_i^{r_i}$, *where* $\sum_i r_i = 1$ *and each* $r_i > 1/\operatorname{poly}$.

Then $f(a,b)$ has a functionally private approximation that can be computed in polynomial time.

For example, in our proof of the private approximability of the permanent, each event is the occurrence of an edge in a random matching of a particular graph, ϕ is a single gate formula consisting of the geometric mean of n inputs, and ψ is the $-n$ power.

Proof. We show that each gate in ϕ with size s below it satisfies the following invariant: If each input takes values in $[1/\operatorname{poly}, 1 - 1/\operatorname{poly}]$, each input can be approximated in polynomial time by sampling, and, for each input, there's a polynomial-time-constructible Bernoulli experiment with success probability indistinguishable from the ideal value, then the output satisfies the same three conditions:

- it takes values in $[1/\operatorname{poly}, 1 - 1/\operatorname{poly}]$,
- it can be approximated in polynomial time by sampling,
- it has associated with it a Bernoulli experiment with success probability indistinguishable from the ideal value.

The first conclusion is clear for each of the gates. The second conclusion follows from the first and third and Lemma 5. As for the third conclusion, we consider the allowed types of gates in turn. We show, for each gate g, that we can construct an experiment with success probability indistinguishable from the output value of g, given coins with success probabilities equal to the input values to g, such that the total number of coins required by g is polynomial. This is easy to do for each gate, using the techniques of Section C.3 where necessary.

As in Lemma 6, it follows that we can estimate f by estimating ϕ and then applying ψ. Also as in Lemma 6, to see that this approximation is functionally private, from $f(a,b)$, a simulator can compute $\psi^{-1}(f(a,b)) = \phi(\Pr(\mathcal{E}_1), \Pr(\mathcal{E}_2), \ldots \Pr(\mathcal{E}_j))$, sample from a distribution with success probability $\psi^{-1}(f(a,b))$, then apply ψ. The result follows.