Secure Multiparty Computation of Approximations

(Extended Abstract)

Joan Feigenbaum[1⋆], Yuval Ishai[2,3], Tal Malkin[3], Kobbi Nissim[4⋆], Martin J. Strauss[3], and Rebecca N. Wright[3]

[1] Yale University, Computer Science Department, New Haven, CT 06520-8285 USA.
[2] DIMACS Center, Piscataway, NJ, USA
[3] AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 USA.
[4] Weizmann Institute, Dept. of Computer Science and Applied Math. Rehovot, Israel.

Abstract. Approximation algorithms can sometimes provide efficient solutions when 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 necessary.
If $\hat{f}$ is an approximation to f, secure multiparty computation of $\hat{f}$ allows the parties to compute $\hat{f}$ without revealing unnecessary information. However, secure computation of $\hat{f}$ may not be as private as secure computation of f, because the output of $\hat{f}$ may itself reveal more information than the output of f. In this paper, we present definitions of secure multiparty approximate computations that retain the privacy of a secure computation of f. We present an efficient, sublinear-communication, private approximate computation for the Hamming distance and an efficient private approximation of the permanent.

1 Introduction

There are an increasing number and variety of real-world applications that collect a massive amount of data and wish to make use of it. 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 is an important emerging area of research [12].

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

⋆ Part of this work was done while the author was at AT&T Labs—Research.

[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.

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.

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 [1,29], recognizing that it is often desirable to perform data mining without revealing unnecessary information about the data.

Separately, each of the above two concerns has been previously addressed. 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 approximate computation 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 [35,18,6,8]) allows a group of parties to compute a function f without revealing unnecessary information.

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—it might be neither 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 an integer-valued 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 important because they give a measure of similarity between two data sets. For example, telephone companies may want to compute joint statistics on their 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, [3,15,21] present algorithms for efficiently approximating L^p distances between two massive data sets. However, these approximations also suffer the kind of information leakage described above. In this paper, our main technical contribution is an efficient $\tilde{O}(n^{1/2})$-communication Hamming distance approximation that does not leak unnecessary information about the parties' inputs.[2]

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 provide a private approximation to the permanent and related #P-hard problems.

Related results. There are several very efficient algorithms for approximating the L^p or Hamming distance (cf. [28,2,15,21]). However, these results do not directly translate into communication-efficient private approximation protocols, as is discussed further in Section 3.

Since the presentation of an early unpublished version of our work [13], Halevi et al. [19] investigate private approximations of NP-hard functions. They show 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 also show that this phenomenon does not hold for all NP-hard functions by presenting an artificial NP-hard function that can be privately approximated.

The approach of constructing private sublinear-communication protocols was initiated in the context of private information retrieval [9] and further studied both in other specific contexts (e.g., [29]) and in more general contexts [31]. 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. Further discussion of other candidate definitions and their drawbacks can be found in [14].

[2] In the full version of this paper [14], we show that if the model is relaxed to allow linear offline communication (before the parties know their inputs), then it is possible to obtain a more efficient solution, with polylogarithmic online communication, for both the Hamming distance and the L^2 distance.

Approximations. In general, an *approximation requirement* could be any binary relation between a deterministic real-valued function f, called the *target function*, and a possibly randomized real-valued function $\hat{f}$, called the *approximating function*, that defines which functions are considered good approximations. The standard approximation relation is the $\langle \epsilon, \delta \rangle$-approximation:

Definition 1. We say that $\hat{f}$ is an $\langle \epsilon, \delta \rangle$*-approximation* to 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 $\hat{f}$.

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. We refer the reader to, e.g., [17,7,4,30] for formal definitions of secure computation in different settings, and to [14] for a somewhat simplified definition that suffices for our purposes. Although in this paper we mostly focus on the simpler case of *privacy*, i.e. security against a passive adversary, our definitions and some of our results also apply to the case of security against an active adversary who can arbitrarily modify the behavior of corrupted parties.

It is known that any polynomial-time computable (randomized) functionality can be privately computed in polynomial time [35,18]. This general feasibility result can be based on the existence of trapdoor permutations, which will be assumed throughout the paper.

Secure Approximations. We restrict our attention to approximating 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. 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, an approximation function $\hat{f}$ is functionally private with respect to the target function, f, if the output of $\hat{f}$ reveals no more information about its input than f does:

Definition 2. Let $f(\mathbf{x})$ be as above and $\hat{f}(\mathbf{x})$ be a possibly randomized function. Then $\hat{f}$ is *functionally t-private with respect to f* if there is an efficient randomized algorithm $\mathcal{S}$, called a *simulator*, such that for every $\mathbf{x}$ and $1 \leq i_1, \ldots, i_t \leq m$, $\mathcal{S}((i_1, x_{i_1}), \ldots, (i_t, x_{i_t}), f(\mathbf{x}))$ 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 $t \geq 0$. 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, 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).

Definition 3. Let f be a deterministic function. The protocol π is a *t-secure $\langle \epsilon, \delta \rangle$-approximation protocol for f* if it t-securely computes a (possibly randomized) function $\hat{f}$, such that $\hat{f}$ is both functionally t-private with respect to f and an $\langle \epsilon, \delta \rangle$-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.

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 latter setting, we will consider sublinear approximations to functions whose exact computation requires at least linear and at most polynomial resources. 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. [20]). Our protocols indeed satisfy these properties, requiring a single pass over the raw data, and sublinear storage, proportional to the communication complexity.

3 Sublinear Private Approximation for the Hamming Distance

In this section we present a two-party private protocol for the approximate Hamming distance. 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 about the other player's input except what follows from the Hamming distance. Our protocol requires roughly $O(n^{1/2})$ bits of communication and three rounds of interaction. By contrast, an exact computation of the Hamming distance requires $\Omega(n)$ communication, even if a small failure probability is allowed [33].

Notation. 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 be read as "$O(x \cdot n^\gamma)$ for an arbitarily small $\gamma > 0$".[3] 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.

Approximations via Sketches. 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 [3,28,11,26]. More specifically, the best $\langle \epsilon, \delta \rangle$-approximations require $O(\log n \log(1/\delta)/\epsilon^2)$ communication. These methods are based on a *sketching* approach:

Definition 4. 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.[4] 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.

We review an efficient sketching protocol for the Hamming distance [28,11]:

Example 5. **(A sketching protocol for the Hamming distance.)** Let the common random input define a 0/1-valued matrix R, with $O(\log n)$ rows and n columns, in which each entry of the i-th row (independently) takes the value

[3] Following the single-parameter security definition of [17], a cryptographic security parameter can be replaced by $\tilde{O}(1)$.

[4] The common random input r can be replaced by a common pseudo-random input, whose succinct representation is picked by one party and communicated to the other.

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 and x are viewed as a matrix and a vector over GF(2), respectively. From the sketches Ra and Rb, the distance $d_h(a, b)$ can be approximated (roughly, by observing that $(Ra)_i = (Rb)_i$ with probability close to 1/2 if $d_h(a, b) \gg 1/p_i$ and with probability close to 1 if $d_h(a, b) \ll 1/p_i$.) The communication complexity of this sketching protocol is $O(\log n)$.

Achieving Privacy. Our goal is to obtain a sublinear-communication *private* approximation protocol for the Hamming distance. A natural approach is to seek 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 5 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. The players exchange a common random input r, locally compute a sketch based on individual inputs 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. Although efficient, this protocol generally fails to be private, even when g is functionally private with respect to f, since knowing the output of g *together with the random input* r *which was used to generate this output* can reveal additional information about the inputs. For instance, in the protocol of Example 5, Alice can deduce Rb from her input a, the output $R(a - b)$, and the common random input R.

We do not know whether the sketching method of Example 5 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. Underlying our solution is a combination of two different sketching protocols with small communication overhead (also referred to as *estimators*) for the Hamming distance. Both these estimators are functionally private and we show how to implement them privately 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 uses a sublinear-communication $\binom{n}{1}$-OT protocol[5] [27,16,34,32,10], which can be based on any homomorphic encryption

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

scheme. The second estimator 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 is private and efficient for any distance.

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. 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 tool is a private protocol for comparing a randomly sampled pair of bits. Formally, the protocol computes $\mathsf{Sample\text{-}XOR}$, defined as $\mathsf{Sample\text{-}XOR}(a, b) = a_r \oplus b_r$, where $r \overset{\mathrm{R}}{\leftarrow} [n]$. Note that a protocol for $\mathsf{Sample\text{-}XOR}$ should keep the choice of r private from each party. Figure 1 describes a private protocol for the function $\mathsf{Sample\text{-}XOR}$ that uses $\binom{n}{1}$-OT as a subprotocol. In it, 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 $x_i \oplus m$.

Private-Sample-XOR

1. Alice picks a random mask $m_A \overset{\mathrm{R}}{\leftarrow} \{0,1\}$ and a random shift amount $r_A \overset{\mathrm{R}}{\leftarrow} [n]$.
   She computes the n-bit string $a' \overset{\mathrm{def}}{=} (a \ll r_A) \oplus m_A$ Symmetrically, Bob picks $m_B \overset{\mathrm{R}}{\leftarrow} \{0,1\}$ and $r_B \overset{\mathrm{R}}{\leftarrow} [n]$, and computes $b' \overset{\mathrm{def}}{=} (b \ll r_B) \oplus m_B$.
2. Alice and Bob invoke in parallel two $\binom{n}{1}$-OT protocols:
   – Alice retrieves $z_A \overset{\mathrm{def}}{=} b'_{r_A}$ from Bob;
   – Bob retrieves $z_B \overset{\mathrm{def}}{=} a'_{r_B}$ from Alice.
3. Alice sends $z'_A \overset{\mathrm{def}}{=} z_A \oplus m_A$ to Bob. Bob sends $z'_B \overset{\mathrm{def}}{=} z_B \oplus m_B$ to Alice.
   Both parties locally output $z'_A \oplus z'_B$.

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

Lemma 6. Private-Sample-XOR *is a private protocol computing the randomized function* Sample-XOR*.*

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 $d_h(a,b)$. Summarizing, we have:

Lemma 7. (Private approximation for the high distance case.) *Let $\mathcal{OT}$ be an arbitrary $\binom{n}{1}$-OT protocol (with security against a passive adversary). 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 $d_h(a,b)$ (i.e., the view of each party can be simulated from its input and $d_h(a,b)$);*
- *If $d = d_h(a,b) \geq d_{\min}$, the protocol outputs an ϵ-approximation of d with overwhelming probability.*

The Low Distance Estimator. We next consider the low distance case, where $d \leq d_{\max}$ for some threshold $d_{\max}$ to be later specified. We design two alternative private protocols for this case, each of which is based on a sketching protocol with 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 $d_h(a,b)$.
- The above value is equal to $d_h(a,b)$ if $d_h(a,b) \leq d_{\max}$ and "fail" otherwise.

For any sketching protocol satisfying the first property above, a private computation of g is straightforward: the parties 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. For such a protocol to be communication-efficient, G must be computable by a small circuit. In [14], we describe two such protocols, one based on hashing and one on Reed-Solomon codes. Using either protocol, we obtain:

Lemma 8. (Private approximation for the low distance case.) *For any $1 \leq d_{\max} \leq n$, there exists a 3-round protocol π with $\tilde{O}(d_{\max})$ communication, such that:*

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

The Combined Protocol. The combined protocol runs both estimators setting $d_{\min} = d_{\max} = n^{1/2}/\epsilon$, and outputs the low distance output if it is not "fail", and the high distance estimator otherwise. Substituting the complexity parameters of the two subprotocols and a 2-round $\tilde{O}(1)$-communication $\binom{n}{1}$-OT protocol yields the following:

Theorem 9. *Assuming the existence of homomorphic encryption, the Hamming distance function can be privately ϵ-approximated with communication complexity $\tilde{O}(n^{1/2}/\epsilon)$ and three rounds of interaction.*

4 Efficient Approximations of #P-hard Functions

We now turn our attention to #P-hard problems, where the goal is to achieve *polynomial-time* private approximations. Note that *artificial* privately approximable #P-hard problems are easily constructed. 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 can be privately approximated by simply outputting 2^{2n}. However, this approximation does not apply to an "interesting" quantity.

In this section, we give a private approximation of the *permanent*, the most well-known #P-complete problem. The permanent $\mathrm{per}(M)$ of a 0/1-valued matrix M counts the number of perfect matchings in the corresponding bipartite graph. Our result extends to other natural #P-complete problems, which reduce to the permanent in an approximation-preserving way. These include counting problems that arise naturally in physics, such as the number of tilings of certain lattices, or problems concerning the bond strength of molecules [25].

Approximating the Permanent. A string of results [22,5,23,24] provides efficient approximation algorithms for the permanent. We build on their technique to achieve a private approximation.

Theorem 10. *Let $f(M_1, M_2) = \mathrm{per}(M_1 + M_2)$, where M_1, M_2 are $n \times n$ matrices with n-bit non-negative entries. Then, for any $\epsilon \geq 1/\mathrm{poly}(n)$ there is a polynomial-time private ϵ-approximation for f.*

Proof (sketch): Let $M = M_1 + M_2$. Using generic secure computation, it suffices to show how to approximate $\mathrm{per}(M)$ in polynomial time in a functionally private way.

Assume that $\mathrm{per}(M) > 0$ (this can be efficiently verified). The technique of [24] uses a rapidly-mixing Markov chain to sample from the set of all perfect matchings on a graph from a distribution that is statistically indistinguishable from uniform. For our purposes, we note only that their techniques provide an efficient implementation of "coins" with success probabilities statistically indistinguishable from $p_1, \ldots, p_n$, where $\prod 1/p_i = \mathrm{per}(M)$, and, for all i, $p_i \geq 1/\mathrm{poly}(n)$. Since each probability can be efficiently estimated by sampling, a (non-private) approximation for the product, and thus for $\mathrm{per}(M)$, follows.

Note, however, that the product of approximations *does* potentially leak information about its factors (e.g., the standard deviation depends on the factors), and, thus, the product of approximations is not functionally private. Fortunately, we are able to avoid this leakage by manipulating success probabilities in various ways. Given coins with success probabilities q_0, q_1 and r, one can form coins with success probabilities $q_0 q_1$, $1 - q_0$, and $rq_1 + (1 - r)q_0$ (the latter by flipping r and then flipping q_r), without knowing any probability.

We proceed, roughly, as follows. First, we observe that if we could construct coins with success probabilities $p_i^{1/n}$, then we could also construct a coin with success probability $\prod p_i^{1/n} = 1/\mathrm{per}(M)^{1/n}$. Since $1 \leq \mathrm{per}(M)^{1/n} \leq \mathrm{poly}(n)$, the success probability of the latter coin can be efficiently estimated, yielding an efficient and functionally-private approximation for the permanent. It thus remains to create coins with success probabilities $p_i^{1/n}$. We use the Taylor series for $x^{1/n}$ at $x = 1$. This series has the form $1 - \sum_{i \geq 1} c_i (1-x)^i$, where $\sum c_i \leq 1$ and each c_i is positive. For $1/\mathrm{poly}(n) \leq x \leq 1$, one can estimate $x^{1/n}$ by the first $kn^{O(1)}$ terms, denoted $T(x)$, with an error negligible in k. We conclude by observing that $T(x)$ is of the form constructible, using the above manipulations, from a coin with success probability x. ∎

Acknowledgements

We thank Dana Randall for suggesting some applications in Section 4 and Jessica Fong for helpful discussions and collaboration in early stages of this work.

References

1. R. Agrawal and S. Ramakrishnan. Privacy-preserving data mining. In ACM SIGMOD, 439–450, 2000.
2. N. Alon, P. Gibbons, Y. Matias, and M. Szegedy, Tracking join and self-join sizes in limited storage. In 18th PODS, 10–20, New York, 1999.
3. N. Alon, Y. Matias, and M. Szegedy, The Space Complexity of approximating the frequency moments. In 28th STOC, 20–29, 1996.
4. D. Beaver, Foundations of secure interactive computing, *CRYPTO '91, LNCS 576,* 1991.
5. A. Broder, How Hard Is It To Marry At Random? (On the approximation of the permanent), In 18th STOC, 1986, 50–58. Erratum in 20th STOC, p. 551.
6. M. Ben-Or, S. Goldwasser, and A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computing. In 20th STOC, 1–10, 1988.
7. R. Canetti, Security and composition of multiparty cryptographic protocols, *Journal of Cryptology,* Vol. 13, No. 1, Winter 2000.
8. D. Chaum, C. Crépeau, and I. Damgård, Multiparty unconditionally secure protocols. In 20th STOC, 11–19, 1988.
9. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In 36th FOCS, 41–51, 1995. Journal version: *J. of the ACM*, 45:965–981, 1998.
10. C. Cachin, S. Micali, and M. Stadler, Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology: EUROCRYPT '99*, LNCS 1592, 402–414, 1999.
11. G. Cormode, M. Paterson, S. Sahinalp, and U. Vishkin, Communication complexity of document exchange. In 11th SODA, 197–206, 2000.
12. *DIMACS special year on massive data sets, 1997–1999,* `http://dimacs.rutgers.edu/SpecialYears/1997\_1998/`.
13. J. Feigenbaum, J. Fong, M. Strauss, and R. Wright, Secure multiparty computation of approximations, presented at *DIMACS Workshop on Cryptography and Intractability*, 2000.

14. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright, Secure multiparty computation of approximations, available at `http://eprint.iacr.org/2001/024/`.
15. J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, An approximate L^1-difference algorithm for massive data streams. In 40th FOCS, 501–511, 1999.
16. 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 30th STOC, 1998.
17. O. Goldreich, *Secure multi-party computation,* (working draft, Version 1.1), 1998. Available from `http://philby.ucsd.edu/cryptolib/BOOKS/oded-sc.html`.
18. O. Goldreich, S. Micali, and A. Wigderson, How to play any mental game. In 19th STOC, pp. 218–229, 1987.
19. S. Halevi, E. Kushilevitz, R. Krauthgamer, and K. Nissim, Private approximations of NP-hard functions. To appear, STOC 2001.
20. M. Rauch Henzinger, P. Raghavan, and S. Rajagopalan. *Computing on data streams*. Technical Report 1998-011, Digital Equipment Corporation Systems Research Center, 1998.
21. P. Indyk, Stable distributions, pseudorandom generators, embeddings and data stream computation. In 41st FOCS, 2000.
22. M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science* **43** (1986) 169–188.
23. M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing* **18** (1989), 1149–1178.
24. M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. STOC 2001, to appear.
25. R. Kenyon. Local statistics of lattice dimers. *Ann. IHP Prob. Stat.* **33** (1997), 591–618.
26. E. Kushilevitz and N. Nisan, *Communication complexity.* Cambridge University Press, 1997.
27. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In 38th FOCS, 364–373, 1997.
28. E. Kushilevitz, R. Ostrovsky, and Y. Rabani, Efficient search for approximate nearest neighbor in high dimensional spaces. In 30th STOC, 614–623, 1998.
29. Y. Lindell and B. Pinkas, Privacy preserving data mining. In *Advances in Cryptology - CRYPTO '00*, LNCS 1880, 36–54, 2000.
30. S. Micali and P. Rogaway, "Secure computation", *CRYPTO '91, LNCS 576,* Springer-Verlag, 1991.
31. M. Naor, and K. Nissim. Communication preserving protocols for secure function evaluation. To appear, STOC 2001.
32. M. Naor and B. Pinkas, Oblivious transfer and polynomial evaluation. In 31st STOC, 245–254, 1999.
33. K. Pang and A. El-Gamal. Communication complexity of computing the Hamming distance. *SIAM J. on Computing*, 15(4):932–947, 1986.
34. J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology - ASIACRYPT '98*, LNCS 1514, 357–371, 1998.
35. A. C. Yao, Protocols for secure computation. In 23rd FOCS, 160–164, 1982.
36. A. C. Yao. How to generate and exchange secrets. In 27th FOCS, 162–167, 1986.