

# Secure Multiparty Computation of Approximations\*

Joan Feigenbaum<sup>†</sup>    Yuval Ishai<sup>‡</sup>    Tal Malkin<sup>§</sup>    Kobbi Nissim<sup>¶</sup>  
Martin J. Strauss<sup>||</sup>    Rebecca N. Wright<sup>\*\*</sup>

## 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 may be extremely large. Furthermore, for some applications, the parties want to compute a function of their inputs *securely*, without revealing more information than necessary. In this work we study the question of simultaneously addressing the above efficiency and security concerns via what we call *secure approximations*.

We start by extending standard definitions of secure (exact) computation to the setting of secure approximations. Our definitions guarantee that no additional information is revealed by the approximation beyond what follows from the output of the function being approximated. We then study the complexity of specific secure approximation problems. In particular, we obtain a sublinear-communication protocol for securely approximating the Hamming distance and a polynomial-time protocol for securely approximating the permanent and related #P-hard problems.

## 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. (For example, see [18] 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 a quadratic algorithm cannot generally be considered practical on inputs consisting of a terabyte of data; such data sets are now routinely generated daily.

In addition to the efficiency of a computation, an important concern 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

---

\*A preliminary version of this work appeared in *Proceedings of 28th International Colloquium on Automata, Languages and Programming (ICALP)*, 2001 [22]. Part of this work was done while all authors were at AT&T Labs—Research. Partial support to the second, fourth, and sixth authors was also provided by DIMACS.

<sup>†</sup>Computer Science Department, Yale University, New Haven, CT 06520 USA. [joan.feigenbaum@yale.edu](mailto:joan.feigenbaum@yale.edu). Supported in part by ONR grant N00014-01-1-0795 and NSF grant CCR-0331548.

<sup>‡</sup>Computer Science Department, Technion, Haifa 32000 Israel. [yuvali@cs.technion.ac.il](mailto:yuvali@cs.technion.ac.il). Supported in part by grant 36/03 from the Israel Science Foundation.

<sup>§</sup>Department of Computer Science, Columbia University, New York, NY 10027 USA. [tal@cs.columbia.edu](mailto:tal@cs.columbia.edu). Supported in part by NSF grant CCF-0347839.

<sup>¶</sup>Department of Computer Science, Ben-Gurion University, Beer Sheva, 84105, Israel. [kobbi@cs.bgu.ac.il](mailto:kobbi@cs.bgu.ac.il). Work partially done while at the Weizmann Institute.

<sup>||</sup>Departments of Math and EECS, University of Michigan, Ann Arbor, MI 48109 USA [martinjs@umich.edu](mailto:martinjs@umich.edu). Supported in part by NSF grant DMS-0354600.

<sup>\*\*</sup>Stevens Institute of Technology, Department of Computer Science, Hoboken, NJ 07030 [rwright@cs.stevens.edu](mailto:rwright@cs.stevens.edu). Supported in part by NSF grant CCR-0331584.

party learns anything about the others' inputs except what is implied by her own output. 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, 44, 14], 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 [55, 29, 9, 15]) allows a group of parties to compute a function  $f$  without revealing unnecessary information.

We address both concerns simultaneously. We construct approximation algorithms that are more efficient than exact computation and that maintain the privacy of the data. Note that the straightforward approach of simply computing an approximation  $\hat{f}$  via a secure multiparty computation, does not work, because even a secure computation of  $\hat{f}$  may leak information through its output. That is, there could be information about players' inputs that is deducible from the output of  $\hat{f}$  that is not deducible from the output of  $f$ . To illustrate this, consider an integer-valued function  $f$  and an approximation  $\hat{f}$  to  $f$  that outputs  $f(x_1, \dots, x_n)$  with the last bit possibly flipped so that last bit is 0 if  $x_1$  is even and 1 if  $x_1$  is odd. Then  $\hat{f}(x_1, \dots, x_n)$  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 secure approximation 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 distributed computation of the Hamming distance and similar distance functions requires linear communication, there has been much recent work on sublinear-communication distance approximations (while maintaining polynomial computation, low storage, and ideally only a single pass over the raw data). For example, several recent papers [3, 23, 32] present algorithms for efficiently approximating  $L^p$  distances between two massive data sets. These approximations, however, suffer the kind of information leakage described above. One of the main technical contributions of this paper is a secure two-party protocol for approximating the Hamming distance between two  $n$ -bit strings, requiring  $\tilde{O}(n^{1/2})$  communication bits. In a relaxed model allowing offline interaction before the parties know their inputs, we also give a secure approximation for the  $L^2$  norm and the Hamming distance, with online communication at most polynomial in  $\log(n) \log(1/\delta)k/\epsilon$ , where  $\delta$  is the failure probability,  $\epsilon$  is a distortion parameter, and  $k$  is a security parameter.<sup>1</sup>

The techniques we use for the Hamming distance protocol have some independently interesting applications to communication complexity. In particular, they allow two parties to decide whether the Hamming distance between their inputs is at most  $d$  by each sending a randomized message of length  $O(d \cdot \text{polylog } d)$  to a referee. This solves an open problem of Yao [56] asking whether communication complexity better than  $O(d^2)$  is possible.<sup>2</sup>

Approximation algorithms are also useful in the setting where the data involved is only moderate in size, but the function to be computed is computationally hard. We also consider this case and provide a secure approximation to a natural and important  $\#P$ -complete problem, the permanent. We further show how our techniques can be applied to a more general class of problems that have known (non-secure) Monte Carlo Markov chain-based approximations.

<sup>1</sup>All logarithms in this paper are base 2, unless otherwise specified.

<sup>2</sup>This application of our techniques was brought to our attention by Ziv Bar-Yossef [5], who, in joint work with T. S. Jayram and Ravi Kumar, has independently obtained a similar solution to Yao's problem.

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

- definitions of secure multiparty approximations;
- a sublinear-communication solution for the Hamming distance;
- polynomial-time solutions to natural #P-hard problems including the permanent.

**Related work.** There are several very communication-efficient algorithms for approximating the  $L^p$  or Hamming distance (e.g., [42, 2, 23, 32]). These results, however, do not translate into efficient secure approximation protocols, as is discussed further in Section 5.

The approach of constructing secure sublinear-communication protocols was initiated in the context of private information retrieval [16] and further studied both in other specific contexts (e.g., [44]) and in more general contexts [50, 14]. In [50], Naor and Nissim present a general methodology for transforming protocols in the communication complexity model into secure protocols with a low communication overhead. However, the secure protocols obtained by applying their methodology to existing (non-secure) low-communication protocols for approximate Hamming distance yield solutions requiring super-polynomial computation.

Following the publication of a preliminary version of our work [22], Halevi et al. [31] consider secure approximations of NP-hard functions and show some negative results. Specifically, 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 do admit good approximation algorithms without the security restriction. They also present a relaxation of our private approximation definition, that allows the leakage of very little information. Under this definition, they demonstrate that every function admitting a deterministic approximation also admits an “almost private” approximation of related quality. In particular, the size of the minimum vertex cover may be approximated within factor 4 leaking a single bit of information.

Freedman, Nissim, and Pinkas gave an efficient private approximation protocol for computing the intersection size of two databases [24]. One of their protocols uses a generalized version of our Private-Sample-XOR protocol (Section 5).

Indyk and Woodruff recently obtained an elegant private protocol for approximating the  $L^2$  distance between two vectors, requiring only a polylogarithmic amount of communication [33]. Their protocol generalizes and improves our  $\tilde{O}(n^{1/2})$ -communication protocol for the Hamming distance (Section 5).

Beimel et al. formalized a more general notion of private approximations that applies to search problems, such as finding an approximate vertex cover in a graph [8]. Their privacy requirement is, roughly, that the approximation algorithm should not enable distinguishing two instances that have (exactly) the same set of optimal solution. Their main result is that even under this seemingly weak requirement, vertex cover and max-3-sat do not admit private approximation.

Another approach to privacy has received increasing attention since the work of Agrawal and Srikant [1]. Here, the input to, or outcome of, a computation is perturbed with the goal of hiding sensitive personal data while allowing some utility. Although perturbation may be a tool in achieving private approximations (according to our definitions), that approach is very different from ours. In Section 4, we discuss the limitations of some types of perturbation—rounding and adding random noise—with respect to our notion of privacy.

**Organization.** We provide background definitions for approximations and secure multiparty computation in Section 2. We give our definitions of secure multiparty approximations in Section 3, with additional discussion in Appendices A and B. In Section 4, we discuss rounding and when it is and is not useful for providing private approximations. We present our main private approximation protocols in Section 5 (Hamming distance and  $L^2$  distance) and Section 6 (#P-hard problems).

## 2 Background

In this section, we present background and notation for approximation and secure multiparty computation. Throughout this paper,  $n$  serves as an input length parameter. We measure the complexity of our protocols, their quality of approximation, and the success of an adversary attacking them as functions of  $n$ .

A function  $f : \mathbb{N} \rightarrow [0, 1]$  is *negligible* if it is asymptotically smaller than any inverse polynomial, i.e.,  $f(n) \in n^{-\omega(1)}$ . The function  $f$  is *overwhelming* if  $1 - f$  is negligible. We use the standard asymptotic notation  $\tilde{O}(\cdot)$  in a slightly nonstandard way. By default, an assertion of the form  $C(n) \in \tilde{O}(c(n))$  should be read as: “ $C(n) \in O(c(n) \cdot n^\gamma)$  for an arbitrarily small constant  $\gamma > 0$ ”. However, it is often the case that the stronger, more standard, assertion  $C(n) \in O(c(n) \cdot \log^{O(1)} n)$  holds. In fact, if our default cryptographic assumptions are replaced by stronger ones, then all of the occurrences of  $\tilde{O}(c(n))$  in this paper can be replaced by  $O(c(n) \log^{O(1)} n)$ . See the discussion after Theorem 2.4 for a more concrete discussion of these assumptions.

A *distribution ensemble*  $D = \{D_x\}_{x \in X}$  is a family of probability distributions indexed by some infinite set  $X$  of binary strings. We sometimes take  $X = \{1^n : n \in \mathbb{N}\}$ , in which case the indices in  $X$  are viewed as natural numbers.

**Definition 1** *Two distribution ensembles  $D = \{D_x\}_{x \in X}$  and  $D' = \{D'_x\}_{x \in X}$  are statistically indistinguishable, (written  $D \stackrel{s}{\equiv} D'$ ), if there is a negligible function  $\mu(\cdot)$  such that, for every  $x \in X$ ,*

$$\text{SD}(D_x, D'_x) < \mu(|x|),$$

where  $\text{SD}$  denotes statistical distance defined by  $\text{SD}(Z, Z') = \frac{1}{2} \sum_a |\Pr(Z = a) - \Pr(Z' = a)|$ .

*Ensembles  $D$  and  $D'$  are computationally indistinguishable, (written  $D \stackrel{c}{\equiv} D'$ ), if for every family  $\{C_n\}$  of polynomial-size circuits there exists a negligible function  $\mu(\cdot)$  such that for every  $x \in X$  of length  $n$ ,*

$$|\Pr(C_n(D_x) = 1) - \Pr(C_n(D'_x) = 1)| < \mu(n).$$

## 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 *approximation function*. The relation  $\mathcal{P}$  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  $f$ . A standard requirement, referred to as  $\langle \epsilon, \delta \rangle$ -approximation, is defined as follows.

**Definition 2** *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  $\hat{f}$ .

In this paper, we primarily refer to  $\langle \epsilon, \delta \rangle$ -approximations. In an  $\langle \epsilon, \delta \rangle$ -approximation, both  $\epsilon$  and  $\delta$  may be functions of the input length parameter  $n$ . We often omit the failure probability  $\delta$ , in which case it should be understood to be negligible.

The following folklore lemma, based on a Chernoff-bound argument, is used in several of our proofs. Informally, it says that if a random variable  $X$  has variance small enough compared with its mean, then the mean of  $X$  can be estimated efficiently through multiple samples of  $X$ . We give the proof for completeness, as we were not able to find a proof of this exact result in the literature.

**Lemma 2.1** *Let  $X$  be a real-valued random variable and suppose that, for some  $c$ ,  $E[X^2] \leq cE^2[X]$ . 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

$$\begin{aligned} \text{var}[Y] &= \epsilon^2/(8c)\text{var}[X] \\ &\leq \epsilon^2/(8c)E[X^2] \\ &\leq \epsilon^2 E^2[X]/8. \end{aligned}$$

By the Chebychev inequality,  $\Pr(|Y - E[X]| > \epsilon E[X]) \leq \text{var}(Y)/\epsilon^2 E^2[X] \leq 1/8$ . Let  $Z$  be the median of  $m = 3 \log(1/\delta)$  independent copies of  $Y$ . Then  $|Z - E[X]| \geq \epsilon E[X]$  iff for at least half the  $Y_i$ 's,  $|Y_i - E[X]| \geq \epsilon E[X]$ . Let  $A_i = 1$  if  $|Y_i - E[X]| \geq \epsilon E[X]$  and  $A_i = 0$  otherwise; let  $A = \sum A_i$ . For each  $i$ ,  $E[A_i] \leq 1/8$ , so  $E[A] = m/8$ , and it follows that

$$\begin{aligned}
\Pr(|Z - E[X]| \geq \epsilon E[X]) &= \Pr\left(A > \frac{m}{2}\right) \\
&= \Pr\left(A > (1+3)\frac{m}{8}\right) \\
&\leq \left[\frac{e^3}{(1+3)^{(1+3)}}\right]^{\frac{m}{8}} \\
&\approx 1.374^{-m} \\
&\leq 2^{-m/3} \\
&= \delta,
\end{aligned} \tag{1}$$

where (1) follows from a version of the Chernoff bound (e.g., see [4]). ■

## 2.2 Secure Multiparty Computation

Secure multiparty computation allows two or more parties to evaluate a specified function of their inputs while hiding their inputs from each other. When formally defining security, it is convenient to think of an adversary that 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 in an *ideal process*, 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.

There are several notions of security with various degrees of strength (e.g., [28, 12, 6, 47]). In this work, we mostly deal with the special case of *private computation*, which assumes that the adversary is *passive* (also called *semi-honest* or *honest-but-curious*) and cannot modify the behavior of corrupted parties. In particular, private computation is only concerned with the information learned by the adversary, and not with the effect misbehavior may have on the protocol's correctness. However, both our general definitions and our results apply also to the case of an *active* (or *malicious*) adversary, who can modify the corrupted parties' behavior arbitrarily. (See some further discussion about this at the end of this section.) In the sequel, we use the term "secure" when the discussion applies to both the active and the passive case, and the term "private" for the passive case only.

Another distinction between different notions of security is the extent to which the transcript produced by the ideal-process adversary should resemble the one produced by the real-life execution of the protocol. The three standard variants are *perfect*, *statistical*, and *computational* indistinguishability. These naturally define corresponding notions of perfect, statistical, and computational security. (In the former two cases, the adversary is assumed to be computationally unbounded, whereas in the latter case it is bounded to polynomial time.) In this work we focus mainly on the two-party case, in which only computational security can be achieved. However, our definitions and some of our results apply to the other variants as well.

We next define private two-party computation, closely following the definition of Goldreich [28]. An extension of this definition to the multiparty case and to the case of security against malicious parties is sketched in Appendix A.

**Functionality.** A two-party computation task is specified by a (possibly randomized) mapping  $g$  from a pair of inputs  $(a, b) \in \{0, 1\}^* \times \{0, 1\}^*$  to a pair of outputs  $(c, d) \in \{0, 1\}^* \times \{0, 1\}^*$ . We refer to such a mapping as a *functionality* (or sometimes simply as a *function*). Without loss of generality, we assume that the inputs  $a, b$  are both of the same length  $n$ ; if this is not the case, padding may be applied. (This convention allows us to use the input length as a security parameter.) We sometimes refer to single-output functionalities, in which case the two outputs of the corresponding two-output functionality  $g$  are assumed to be identical.

**Protocol.** A two-party protocol is defined by a pair of probabilistic polynomial-time interactive algorithms  $\pi = (\pi_A, \pi_B)$ . The protocol  $\pi$  is executed as follows. Initially, Alice, who operates according to  $\pi_A$ , receives an input  $a$  and a random input  $r_A$ , and Bob, who operates according to  $\pi_B$ , receives an input  $b$  and a random input  $r_B$ . We assume that  $|a| = |b| = n$ . The execution then proceeds by synchronous rounds, where, at each round, each party may send to the other party a message as specified by  $\pi$ , based on her input, her random input, and messages received in previous rounds. At each round, each party may decide to terminate and output some value based on her entire view (consisting of her input, random input, and received messages).

**Private computation.** For defining the privacy of  $\pi$  with respect to a functionality  $g$ , it is convenient to use the following notation. Consider the probability space induced by the execution of  $\pi$  on input  $\mathbf{x} = (a, b)$  (induced by the independent choices of the random inputs  $r_A, r_B$ ). Let  $\text{view}_A^\pi(\mathbf{x})$  (resp.,  $\text{view}_B^\pi(\mathbf{x})$ ) denote the entire view of Alice (resp., Bob) in this execution, including her input, random input, and all messages she has received. Let  $\text{output}_A^\pi(\mathbf{x})$  (resp.,  $\text{output}_B^\pi(\mathbf{x})$ ) denote Alice's (resp., Bob's) output. Note that the above four random variables are defined over the same probability space.

**Definition 3** Let  $X$  be the set of all valid inputs  $\mathbf{x} = (a, b)$  (i.e., pairs of equal-length binary strings). A protocol  $\pi$  is a private protocol computing  $g$  if the following properties hold:

**Correctness.** The joint outputs of the protocol are distributed according to  $g(a, b)$ . Formally,

$$\{(\text{output}_A^\pi(\mathbf{x}), \text{output}_B^\pi(\mathbf{x}))\}_{\mathbf{x} \in X} \equiv \{(g_A(\mathbf{x}), g_B(\mathbf{x}))\}_{\mathbf{x} \in X},$$

where  $(g_A(\mathbf{x}), g_B(\mathbf{x}))$  is the joint distribution of the outputs of  $g(\mathbf{x})$ .

**Privacy.** There exist probabilistic polynomial-time algorithms  $\mathcal{S}_A, \mathcal{S}_B$ , called simulators, such that:

$$\begin{aligned} \{(\mathcal{S}_A(a, g_A(\mathbf{x})), g_B(\mathbf{x}))\}_{\mathbf{x}=(a,b) \in X} &\stackrel{c}{\equiv} \{(\text{view}_A^\pi(\mathbf{x}), \text{output}_B^\pi(\mathbf{x}))\}_{\mathbf{x} \in X} \\ \{(g_A(\mathbf{x}), \mathcal{S}_B(b, g_B(\mathbf{x})))\}_{\mathbf{x}=(a,b) \in X} &\stackrel{c}{\equiv} \{(\text{output}_A^\pi(\mathbf{x}), \text{view}_B^\pi(\mathbf{x}))\}_{\mathbf{x} \in X} \end{aligned}$$

The above privacy requirement asserts that whatever the real-life adversary learns by (passively) corrupting a party, an ideal-process adversary can simulate by only learning the input and output of that party. Note that the definition does not consider the view of the corrupted party alone, but rather concatenates this view to the output of the uncorrupted party. When the functionality  $g$  is randomized, this serves to ensure that the adversary does not learn additional information about the output of the other party; e.g., via correlations present in the real-life process but absent in the ideal process.

**Protocol composition.** In order to design and analyze private protocols in a modular way, it is convenient to rely the following composition paradigm. First, design a high-level *oracle-aided* protocol  $\pi^{f|g}$ , which is a private protocol for  $f$  in which the parties may call a trusted party (oracle) that computes a (presumably simpler) functionality  $g$ . Then, substitute each oracle call in  $\pi^{f|g}$  by an invocation of a private protocol  $\pi^g$  computing  $g$ . This approach is justified by the following composition theorem:

**Theorem 2.2** [12, 28] Let  $f, g$  be (deterministic or randomized) two-party functionalities. Let  $\pi^{f|g}$  be a private oracle-aided protocol for  $f$  which uses oracle calls to  $g$ , and let  $\pi^g$  be a private protocol for  $g$ . Let  $\pi^f$  be the protocol obtained from  $\pi^{f|g}$  by independently invoking  $\pi^g$  for implementing each oracle call to  $g$ . Then  $\pi^f$  is a private protocol for  $f$ .

**Feasibility results.** The first general feasibility results for secure computation were obtained by Yao [55] and by Goldreich, Micali, and Wigderson [29]. The following theorem relates the complexity of privately computing a functionality  $g$  to the circuit size of  $g$ .

**Theorem 2.3** [55] Let  $C = \{C_n\}$  be a uniform family of (deterministic or probabilistic)<sup>3</sup> Boolean circuits of size  $s(n)$ ,  $s(n) \geq n$ , where the input to  $C_n$  is viewed as a pair of  $n$ -bit strings and its output as a pair of strings. Let  $g$  denote the functionality computed by the family  $C$ . Then, assuming the existence of enhanced

<sup>3</sup>A probabilistic circuit includes, in addition to the standard inputs, a polynomial number of random inputs.

trapdoor permutations<sup>4</sup> (resp., homomorphic encryption schemes<sup>5</sup>),  $g$  can be privately computed in three (resp., two) rounds with  $\tilde{O}(s(n))$  bits of communication.

Theorem 2.3 can be generalized (with a larger, but still constant, number of rounds) also to the case where the adversary is active [43] and to computations involving more than two parties [7, 39, 52]. In the case of constant-round multiparty computation with no honest majority, one needs to additionally assume the existence of collision-resistant hash functions.

A particularly useful private computation task is that of *oblivious transfer* [53, 20], defined below.

**Definition 4 (Oblivious Transfer)** *An  $n$ -choose-1 oblivious transfer protocol (with security against a passive adversary), abbreviated as  $\binom{n}{1}$ -OT, is a private protocol for the following deterministic functionality between two parties: a sender and a receiver. 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.*

By Theorem 2.3,  $\binom{n}{1}$ -OT can be implemented with nearly linear communication. However, the  $\binom{n}{1}$ -OT functionality also admits much more efficient solutions:

**Theorem 2.4** [41, 27, 54, 46, 51] *Assuming the existence of a homomorphic encryption scheme, there is a 2-round  $\binom{n}{1}$ -OT protocol with  $\tilde{O}(1)$  bits of communication.*

**On complexity vs. assumptions.** As noted at the start of this section, the asymptotic complexity notation  $\tilde{O}(c(n))$  should be read by default as  $O(c(n) \cdot n^\gamma)$  for an arbitrarily small constant  $\gamma > 0$ . In both Theorem 2.3 and Theorem 2.4,  $\tilde{O}(c(n))$  can be read as  $O(c(n) \cdot \log^{O(1)} n)$  if stronger cryptographic assumptions are made. Specifically, in Theorem 2.3 it suffices to assume enhanced trapdoor permutations (or homomorphic encryption) secure against sub-exponential adversaries, and in Theorem 2.4 it suffices to assume specific number-theoretic assumptions from [11, 26] or [45]. (The latter assumptions are required for implementing  $\binom{n}{1}$ -OT with polylogarithmic communication.) Because the efficiency improvement resulting from the stronger assumptions would not be very significant for our purposes, we use the more conservative assumptions by default.

**On passive vs. active adversaries.** For simplicity, we mainly focus in this paper on the case of security against a passive adversary. Using a theorem from [50], this does not compromise generality: any protocol with security against a passive adversary can be upgraded into a protocol with security against an active (or “malicious”) adversary with only a small overhead to the communication complexity. Thus, our results are quite insensitive to the distinction between the two types of adversaries.

### 3 Secure Approximations

In this section, we present our definition of 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  $\mathbf{x} = (x_1, \dots, x_m) \in X$  to a non-negative number  $y$ . Each string  $x_i$  is the input held by the  $i$ th party. As before, all inputs  $x_i$  are assumed to have the same length.

We start by defining a notion of *functional privacy* on which our main definition relies. 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 about its input than  $f$  does. Note that this is an inherent property of the function  $\hat{f}$  rather than of a particular protocol computing  $\hat{f}$ . The notion of functional privacy is formally defined as follows.

<sup>4</sup>See Appendix C.1 of [28] for a definition of enhanced trapdoor permutations.

<sup>5</sup>Loosely speaking, a semantically secure encryption scheme [30] is said to be *homomorphic* if: (1) The plaintexts are taken from some group  $(H, +)$ ; (2) From encryptions of group elements  $h_1, h_2$  it is possible to efficiently compute a random encryption of  $h_1 + h_2$ . Homomorphic encryption can be based on a variety of intractability assumptions, including the Quadratic Residuosity Assumption and the Decisional Diffie-Hellman assumption.

**Definition 5 (functional privacy)** Let  $f(\mathbf{x})$  be as above, and let  $\hat{f}(\mathbf{x})$  be a possibly randomized function. We say that  $\hat{f}$  is perfectly (resp., statistically, computationally) functionally private with respect to  $f$  if there exists a probabilistic sampling algorithm  $\mathcal{S}$ , running in expected polynomial time, such that for every input  $\mathbf{x} \in X$ , the distribution  $\mathcal{S}(f(\mathbf{x}))$  is perfectly (resp., statistically, computationally) indistinguishable from  $\hat{f}(\mathbf{x})$ . In the statistical and computational cases, the sampling algorithm  $\mathcal{S}$  is additionally given a security parameter  $n$  and the indistinguishability is defined with respect to this  $n$ .

Our definition for secure approximation requires that the protocol securely compute some functionally private approximation  $\hat{f}$  of  $f$ . Because we defined  $\hat{f}$  to be a single-output function, we must fix some convention for extending it to a multi-output function. As in the two-party case, 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. Here, we prefer simplicity over generality.<sup>6</sup> 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 indistinguishability).

**Definition 6 (secure approximation)** Let  $f$  be as above. The protocol  $\pi$  is a secure  $\mathcal{P}$ -approximation protocol for  $f$  if it securely computes some (possibly randomized) function  $\hat{f}$ , such that  $\hat{f}$  is both functionally private with respect to  $f$  and a  $\mathcal{P}$ -approximation of  $f$ . The type of functional privacy (perfect, statistical, or computational) should match the required type of security.

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. Secure 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. For the former setting, the following simple corollary of Theorem 2.3 and Definition 6 is useful:

**Theorem 3.1** Suppose that  $f$  admits a functionally private  $\mathcal{P}$ -approximation  $\hat{f}$  that can be computed in probabilistic polynomial time. Then  $f$  admits an efficient private  $\mathcal{P}$ -approximation protocol (i.e., a protocol with  $\text{poly}(n)$  communication and computation).

We stress again that Theorem 3.1 only addresses the feasibility of secure approximations and does not deal with more refined efficiency goals such as achieving sublinear communication complexity. Thus, given Theorem 3.1, the design of secure approximation protocols can involve two distinct types of challenges:

- For polynomial-time computable functions  $f$ , the challenge is to design special-purpose protocols that outperform the efficiency of the generic approach. This is the focus of Section 5.
- For intractable functions  $f$ , the challenge is to find an efficiently computable and functionally private approximation  $\hat{f}$  that can be used for applying Theorem 3.1. This algorithmic question is the focus of Section 6.

### 3.1 An Alternative Definition

We now describe a more liberal alternative to the above definition, which is useful for some of our protocols. To motivate the alternative definition, consider an artificial protocol  $\pi$  which first invokes some secure protocol for exactly computing  $f$ , and then instructs each party to output some functionally private approximation  $\hat{f}$  which is computed from the output of  $f$ . Should  $\pi$  be considered a secure approximation protocol for  $f$ ?

---

<sup>6</sup>Other natural alternatives include all players getting independent outputs from the same distribution, or a single distinguished player getting the output and all other players getting nothing. We note that any private approximation protocol for  $f$  under the latter convention (single output) can be easily turned into a private approximation protocol for  $f$  under the other two conventions: in the case of identical outputs it suffices for the distinguished player to send its output to other players, and in the case of independent outputs it suffices to run the protocol multiple times, alternating the role of the distinguished player between the players.



According to Definition 6,  $\pi$  generally cannot be considered secure, as the value of  $f$  learned by the parties may reveal strictly more information than the value of  $\hat{f}$  computed by  $\pi$ . However, it seems reasonable to allow the protocol messages in a secure approximation of  $f$  to tolerate the privacy loss implied by an exact computation of  $f$ , as the functional privacy bound already allows that much leakage. The fact that a higher level of privacy can sometimes be achieved for the protocol’s messages than for its output when settling for an approximate computation of  $f$  should not necessarily be turned into a requirement.

The above discussion gives rise to the following definition. For simplicity, we first formulate the definition for the case of private 2-party computation, modifying Definition 3, and then discuss the general case.

**Definition 7 (private approximation: liberal definition)** *Let  $f$  be a deterministic functionality mapping two inputs to a single output. A 2-party protocol  $\pi$  is a private  $\mathcal{P}$ -approximation protocol for  $f$  in the liberal sense if there exists a functionally private  $\mathcal{P}$ -approximation  $\hat{f}$  such that the following requirements hold:*

**Correctness.** *The joint outputs of the protocol are distributed according to  $(\hat{f}(\mathbf{x}), \hat{f}(\mathbf{x}))$  (where the two outputs of  $\hat{f}$  are identical rather than independent).*

**Privacy.** *There exist probabilistic polynomial-time algorithms  $\mathcal{S}_A, \mathcal{S}_B$ , such that:*

$$\begin{aligned} \{(\mathcal{S}_A(a, f(\mathbf{x}), \hat{f}(\mathbf{x})), \hat{f}(\mathbf{x}))\}_{\mathbf{x}=(a,b) \in X} &\stackrel{c}{\equiv} \{(\text{view}_A^\pi(\mathbf{x}), \text{output}_B^\pi(\mathbf{x}))\}_{\mathbf{x} \in X} \\ \{(\hat{f}(\mathbf{x}), \mathcal{S}_B(b, f(\mathbf{x}), \hat{f}(\mathbf{x})))\}_{\mathbf{x}=(a,b) \in X} &\stackrel{c}{\equiv} \{(\text{output}_A^\pi(\mathbf{x}), \text{view}_B^\pi(\mathbf{x}))\}_{\mathbf{x} \in X}. \end{aligned}$$

*Again, distinct occurrences of  $\hat{f}$  in each of the above expressions are assumed to take the same value.*

Note that in the above definition,  $\hat{f}(x)$  is given as an additional input to  $\mathcal{S}_A$  and  $\mathcal{S}_B$ . This is not needed when  $\hat{f}$  is deterministic, because in this case  $\hat{f}(x)$  can be computed based on  $f(x)$ . However, in the typical case where  $\hat{f}$  is randomized, this is needed in order to properly correlate the simulator’s output with that of the uncorrupted party.

The above definition addresses the special case of secure two-party computation in the presence of a passive adversary. We now turn to the more general case. Our general formulation of the liberal definition can be viewed as a natural relaxation of the standard simulation-based framework for defining secure (exact) computation, as described in Appendix A. Similar to the standard case, we compare the interaction of the real-life adversary with the real protocol to the interaction of an ideal-process adversary with an ideal function evaluation process involving a trusted party. In the standard definition (Definition 10 in Appendix A), the trusted party receives an input from each party and sends the value  $f(\mathbf{x})$  to all parties. In the liberal definition of secure approximations, the trusted party also computes and sends the value of some functionally private  $\mathcal{P}$ -approximation  $\hat{f}(x)$ . All uncorrupted parties output the approximate value  $\hat{f}(x)$ , whereas the exact value  $f(x)$  is only used by the ideal-process adversary to produce a simulated transcript. (This should be contrasted with our default definition of secure approximations, in which the adversary is only given  $\hat{f}(x)$  and not  $f(x)$ .) See Appendix B for a more formal treatment of the liberal definition in a general setting for secure computation.

**Comparing the Two Definitions.** While the results of this paper are quite insensitive to the distinction between the two definitions presented above, it is still instructive to compare the two and justify our choice of the stricter definition as the default one.

The main advantage of the default definition is that it uses the standard notion of exact secure computation as a black box, and can thus be applied in conjunction with any possible definition of security. A second advantage is more subtle and applies only to the case where the function  $f$  is intractable. The liberal definition, in its general form, allows the ideal-process adversary to interact with a trusted party which computes the exact value of  $f$ . Moreover, in the case of security against an active adversary, the ideal-process adversary may choose its inputs to the computation of  $f$  based on its view of the original inputs  $x$ . Thus, it effectively gains (a restricted) oracle access to an intractable function  $f$ . In contrast, the default definition only allows the ideal-process adversary to learn the value of an efficiently computable function  $\hat{f}$ . This distinction appears to be blurred by the fact that in defining the functional privacy requirement for  $\hat{f}$ , the sampling algorithm  $\mathcal{S}$  is given access to the exact value of  $f$ . However, towards producing  $\hat{f}(x)$ , the

sampling algorithm  $\mathcal{S}$  is only allowed to learn the value of  $f$  on the same input  $x$ , rather than on an input  $x'$  which it can control.

The latter disadvantage of the liberal definition is not very significant. First, it only applies in the case of an intractable function  $f$ . Second, even if the simulator is given some extra computational power, one still gets a meaningful security guarantee. This is particularly true in our case, where the outputs of the functions being computed are real numbers rather than cryptographic or other “computational” objects. Finally, it is possible to avoid this disadvantage altogether by applying the following restriction to the liberal definition. Instead of allowing the simulator learn (via the trusted party) the output of  $f$ , one could only allow it to learn the output of some function  $f'$  which is both functionally private with respect to  $f$  and polynomial-time computable. (As before, the simulator also learns the output of  $f$ .) In this variant of the liberal definition, the simulator is not given a significant computational advantage even when  $f$  is intractable.

We note that our default definition is strictly stronger than the liberal definition: if  $\pi$  securely  $\mathcal{P}$ -approximates  $f$  then it also does so in the liberal sense, while the converse of this statement is not true in general. Except where indicated, the positive results obtained in the remainder of this paper all apply to the default definition, and hence also to the liberal definition. However, our main protocol of Section 5 takes a simpler and more natural form under the liberal definition.

## 4 Rounding and Precision

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 work in general. 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 can provide functional privacy, but is not generally efficient. We then show that finite-precision approximations to real-valued functions can be done in a way that provides functional privacy. This is important because many common functions are naturally described as symbolic, real-valued functions but are implemented in finite precision—an inherent approximation. We give a general theorem showing that if such a function can be implemented efficiently in finite precision, then it can be implemented efficiently and privately in finite precision.

**Rounding.** Consider taking an approximation  $\hat{f}$  for  $f$  that is good to within  $(1 \pm \epsilon/3)$  with high probability and rounding it down to a power of  $(1 + \epsilon/3)$ , obtaining a modified approximation  $\hat{g}$ . Then  $\hat{g}$  is in the range  $(1 \pm \epsilon)f$  with high probability, so  $\hat{g}$  is also a good approximation to  $f$ . Intuitively, much of the information in the least significant bits of  $f$  have been lost in the rounding process, so one might hope that  $\hat{g}$  is functionally private with respect to  $f$ . We now show that is not the case.

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  $f(\mathbf{x}) = f(\mathbf{x}')$  and  $x_1 = x'_1$ . Because  $(x_1, f(\mathbf{x})) = (x'_1, f(\mathbf{x}'))$ , if  $\hat{h}$  is to be functionally private with respect to  $f$ , it is necessary that  $\hat{h}(\mathbf{x})$  and  $\hat{h}(\mathbf{x}')$  have indistinguishable distributions. In general, however, the approximations  $\hat{f}(\mathbf{x})$  and  $\hat{f}(\mathbf{x}')$  may have distinguishable distributions. That is, for one or more  $t$ ,  $\Pr(\hat{f}(\mathbf{x}) < t) \neq \Pr(\hat{f}(\mathbf{x}') < t)$ . Furthermore, 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  $\hat{g}$ , which is  $\hat{f}$  rounded down to a power of  $(1 + \epsilon/3)$ , is not functionally private with respect to  $f$ . Thus, in general, rounding does 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 private 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  $f$  that is statistically functionally private with respect to  $f$ .

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 or if  $f$  is easy to compute, this procedure is a simple and straightforward solution. In the remainder of this section, we exploit this solution when we consider an  $f$  that is intuitively easy to compute (in the appropriate model).

**Finite-Precision Approximations to Real-Valued Outputs.** If  $f$  is a discrete-valued function, then  $f$  has exact finite-precision implementations and all such implementations are functionally private with respect to  $f$ . We now consider the case of real-valued symbolic functions, for which the situation is a bit more complicated.

Some approximation algorithms are most naturally described using real-valued functions for intermediate values or outputs. For example, in the approximation of Section 5.4, the output is a median of means of numbers of the form  $(\sum_i s_i(a_i - b_i))^2$ , where  $\langle a_i \rangle$  and  $\langle b_i \rangle$  are inputs and each  $s_i$  is a unit Gaussian-distributed random variable. The functional privacy of that approximation depends on the fact that  $D_1 = \sum_i a_i s_i$  and  $D_2 = \sqrt{\sum_i a_i^2} s_0$  are identically distributed, where  $s_0$  is also a unit Gaussian random variable. To the extent that the  $s_i$ 's are not true Gaussians (due to rounding), the distributions of  $D_1$  and  $D_2$  are not identical—not even computationally indistinguishable, in general. One might worry that functional privacy is thereby destroyed. More generally, one might worry that, given a simple symbolic mathematical function  $f$ , the straightforward finite-precision implementations of  $f$  are not functionally private with respect to  $f$ , or, worse, that  $f$  may not have any computable functionally private implementation at all, even allowing high cost. We now show that the approximation relation resulting from finite-precision approximations to efficiently computable mathematical functions can always be made private by adding noise. This means that, when designing protocols, one can continue to work with functions symbolically, if desired; the finite-precision implementation can be made private automatically. We give a self-contained example and state a theorem for a simple function in the additive approximation model; the techniques generalize to other functions and other models.

Consider the function  $f(x, y) = \log(xy)$ , where, in this section, the logarithm is to the base 10. Then, as a symbolic statement,  $f(1, 10) = f(2, 5) = 1$ . Now consider the following protocol: Alice computes a finite-precision approximation  $L(x)$  to  $\log(x)$ , Bob computes a finite-precision approximation  $L(y)$  to  $\log(y)$ , and they output  $g(x, y) = L(x) + L(y)$ . In many straightforward real-world implementations,  $g(2, 5) \neq g(1, 10)$  even though  $f(2, 5) = f(1, 10)$ , so the function computed by  $g$  is not functionally private with respect to  $f$ ; in practical terms, an adversary can undesirably distinguish between the inputs  $(1, 10)$  and  $(2, 5)$  because  $g(1, 10)$  is always exactly 1 whereas  $g(2, 5)$  often has roundoff error. That is, a straightforward finite-precision computation of  $f$  (an “exact computation” in the finite-precision sense) is not functionally private with respect to  $f$  as a symbolic function.

To remedy this, as above, we exploit the real-valued exact computability of  $f$ , meaning, for any  $\epsilon$ , one can compute  $\log(x) \pm \epsilon$  in time  $(|x| + \log(1/\epsilon))^{O(1)}$ . Then, to compute a private finite-precision approximation to  $f(x, y)$ , proceed as follows. Given security parameter  $k$ , compute  $f(x, y) \pm (\epsilon/3)2^{-k}$ , then add uniformly random noise in the range  $\pm\epsilon/3$ . As in the previous discussion on adding random noise, this gives statistically indistinguishability output on inputs  $(1, 10)$  and  $(2, 5)$ . In this situation, because the log is “exactly computable,” the cost to compute the output is just polynomial in  $k$ , as desired.

A final consideration is that Definition 5 technically does not apply to real-valued functions, as it does not make sense to say that a discrete-input simulator  $S$  takes, as input, the output of a real-valued function  $f$ . In general, we have the following theorem, whose proof follows from the foregoing discussion.

**Theorem 4.1** *Let  $f$  be a multivariate function from integers to the reals with short symbolic description. Suppose, for any integer  $k$  and any  $\mathbf{x}$ , one can compute a value  $\hat{f}(\mathbf{x}) = f(\mathbf{x}) \pm 2^{-k}$  in time  $(|\mathbf{x}| + k)^{O(1)}$ . Then there exists a function  $g$ , from integers to finite-precision reals (i.e., integer multiples of fixed small unit), such that the following properties hold.*

1. (Good approximation.) For all  $\mathbf{x}$ ,  $g(\mathbf{x}) = f(\mathbf{x}) \pm 2^{-k}$ .
2. (Efficiency.)  $g(\mathbf{x})$  is computable in time  $(|\mathbf{x}| + k)^{O(1)}$ .

3. (Functional privacy, in a modified sense.) There is a simulator,  $S$ , such that, for any family  $\{\rho_j\}$  rounding functions that take real values to finite-precision real values satisfying  $|\rho_j(x) - x| \leq 2^{-j}$ , we have  $S(\rho(f(\mathbf{x}))) \stackrel{s}{\equiv} g(\mathbf{x})$ .

Thus any real-valued exact computation can be made statistically functionally private. This parallels the discrete situation in which any discrete-valued exact computation is trivially automatically perfectly functionally private.

## 5 Sublinear Private Approximation for the Hamming Distance

In this section, we present a private two-party protocol for computing approximate Hamming distance. We also give sublinear-communication protocols for related problems. The Hamming distance protocol allows Alice, holding an input  $a \in \{0, 1\}^n$ , and Bob, holding  $b \in \{0, 1\}^n$ , to learn an  $\epsilon$ -approximation of the Hamming distance between  $a, b$  (with a negligible failure probability  $\delta$ ), without learning additional information about the other party's input beyond what follows from the Hamming distance. Our protocol requires roughly  $O(n^{1/2})$  bits of communication and three rounds of interaction. Throughout this section, 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  $x$ .

Before we describe our private protocol, it is instructive to consider the non-private variant of the problem. We first briefly survey known communication-efficient solutions, and then explain why a naive attempt to make those solutions private fails. There are several known methods for approximating the Hamming distance using polylogarithmic communication [3, 42, 17, 40]. More specifically, the best  $(\epsilon, \delta)$ -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 8** A sketching protocol for a 2-argument function  $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbb{N}$  is defined by:

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

On inputs  $a, b \in \{0, 1\}^n$ , 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 as above is said to  $(\epsilon, \delta)$ -approximate  $f$  if  $g$   $(\epsilon, \delta)$ -approximates  $f$ .

Clearly, the communication complexity of a sketching protocol is proportional to the sketch size.

**Remark.** In the above definition and in the following, it is convenient to assume that the parties share a polynomially long common random input string. This assumption can be dispensed with at a low cost using pseudorandomness, as is done in our protocols.

In this paper, we only consider linear sketching functions, i.e., such that  $S(ax + by) = aS(x) + bS(y)$ , where  $x$  and  $y$  are vectors and  $a$  and  $b$  are scalars, and arithmetic is performed over a finite field or the reals. As a special case,  $S(x)$  may select a sample of the positions in  $x$  (where the selected positions are independent of  $x$ ). Furthermore, in this paper,  $G(S_1, S_2)$  always takes the form  $G'(aS_1 + bS_2)$ , and we sometimes refer to the single-input function  $G'$  as the reconstruction function.

We briefly review an efficient sketching protocol for the Hamming distance [42, 17].

**Example 1 (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 1 with probability  $p_i = \beta^i$  for some constant  $\beta$  depending on  $\epsilon$ . The sketching function is defined by  $S(x, R) = Rx$ , where  $R$  and  $x$  are viewed as a matrix and a vector over  $\text{GF}(2)$ , respectively. From the sketches  $Ra$  and  $Rb$ , the distance  $d_h(a, b)$  can be approximated. (The main observation is 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  $d_h(a, b) \ll 1/p_i$ .) More generally, an  $(\epsilon, \delta)$ -approximation can be obtained using a matrix  $R$  with  $O(\log n \log(1/\delta)/\epsilon^2)$  rows. The communication complexity of this sketching protocol is  $O(\log n \log(1/\delta)/\epsilon^2)$  assuming a common random input is available.

Our goal is to obtain a sublinear-communication private approximation protocol for the Hamming distance. A natural approach is to seek a general method for converting an efficient sketching protocol approximating a function  $f$  into a private protocol approximating  $f$ .

Suppose that the randomized function  $g$  induced 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 other sketching protocols for the Hamming distance proposed in the literature. To approximate  $f$  privately, it suffices to let the parties privately compute the randomized function  $g$ . By Theorem 2.3, a general-purpose private computation protocol can be used to evaluate  $g$ . (Note that the randomness used for sketching is considered here as part of the input to  $g$ .) However, the communication complexity of this protocol is at least linear in  $n$ , while we would like to obtain a sublinear-communication private protocol for  $g$ .

At first glance, the following straightforward protocol seems to work. The parties locally compute a sketch based on individual inputs and their common random input  $r$ , and then apply a general-purpose private computation protocol to evaluate  $g = G(s_A, s_B)$  from the sketches  $s_A, s_B$ . By Theorem 2.3, if the sketches are short and  $G$  is not too complex, then the entire protocol can be implemented with sublinear communication. This protocol, however, generally fails to be private. This is due to the fact that although  $g$  is functionally private with respect to  $f$ , the pair  $(g, r)$  (where  $r$  is the random input for  $g$ ) is not functionally private with respect to  $f$ .

We illustrate this difficulty for the case of privately approximating the Hamming distance using the specific sketching method from Example 1. Consider a protocol that first computes  $R(a - b)$  securely, then computes  $g$  from  $R(a - b)$  in the clear. Note that the function  $h : (a, b) \mapsto R(a - b)$  is functionally private with respect to  $(a, b) \mapsto d_h(a, b)$ , but knowing the output of  $h$  together with the random input  $R$  (which was used to generate this output) can reveal additional information about the inputs. For instance, in the above protocol for computing  $h$ , 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  $R, y$  such that  $R$  is distributed as in Example 1 and  $Rb = y$  holds with overwhelming probability. (For instance, given that  $a = 0$ ,  $b = e_i$ , and  $d_h(a, b) = 1$ ,  $y$  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 cannot be simulated in the ideal process, and thus the naive solution fails.<sup>7</sup>

We do not know whether the sketching method of Example 1 can be made private with sublinear communication, nor were we able to obtain a private protocol from any other efficient protocol for approximating the Hamming distance appearing in the literature. Instead, we design a new sketching protocol, whose induced randomized approximation  $g$  can be privately computed with sublinear communication.

Our solution is based on a combination of two different sketching protocols, also referred to as *estimators*. The first estimator is based on sampling, and gives a good approximation only when the distance is high. We provide a special-purpose low-communication private protocol for computing this estimator. At its heart is a special-purpose private protocol for comparing the bits in a random location, which may be of independent interest. The second estimator gives a good approximation only when the distance is low, and, in fact, produces an exact result in this case. We provide two alternative implementations for this estimator, one based on two-level hashing and one based on Reed-Solomon codes. In either case, the output of the low distance estimator is such that even when taken together with the randomness  $r$ , no information is revealed except what follows from the Hamming distance. Thus, in this case, we can use general purpose private computation, as in the naive approach described above, without loss of privacy.

In Sections 5.1 and 5.2, we describe each of the two private estimators separately; we combine them to obtain the final protocol in Section 5.3. We consider the  $L^2$  distance in a relaxed model in Section 5.4.

<sup>7</sup>It turns out that, similarly, even if the parties use general-purpose secure computation to evaluate  $g = G(s_A, s_B)$  from  $s_A, s_B$  without revealing  $R(a - b)$ , it is generally impossible, knowing only  $a$  and  $d_h(a, b)$ , to generate  $R, g$  with  $R$  distributed as above and  $g = G(s_A, s_B)$ . Intuitively, some information about  $Rb$  leaks into  $g = g(R(a - b)) = g(Rb)$ , and, because  $Rb$  is sensitive, so is  $g$  in general.

## 5.1 The High Distance Estimator

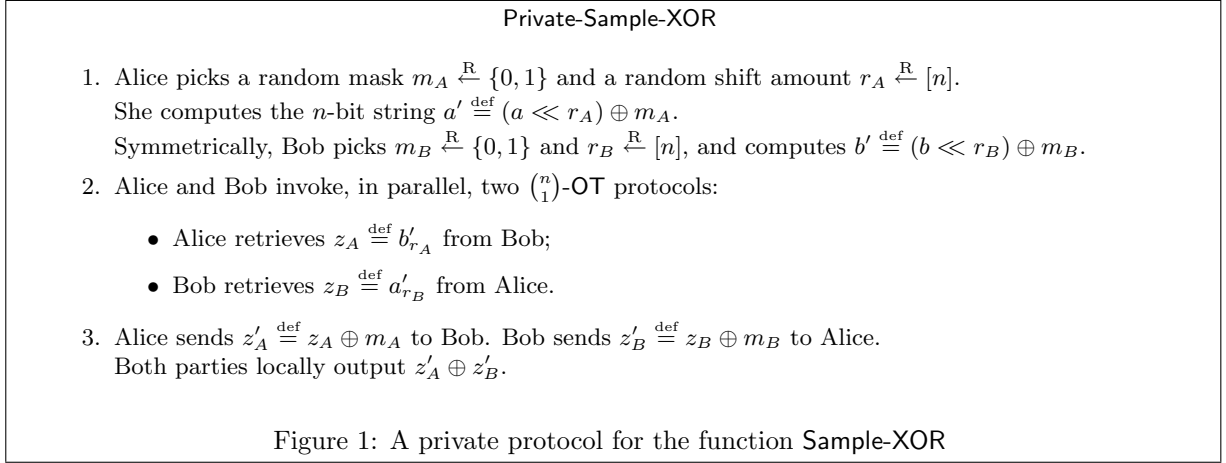
Suppose that  $d = d_h(a, b)$  is guaranteed to be larger than some threshold  $d_{\min}$ . (We will specify later the value we use for  $d_{\min}$ ). If  $d_{\min}$  is large, then Alice and Bob can efficiently approximate  $d$  by randomly sampling a small number of bits in matching positions from their inputs. Viewed as a simple sketching protocol, the common 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  $\Delta$  of differences in  $s = O((n/d_{\min}) \cdot \log(1/\delta)/\epsilon^2)$  randomly selected matching bits of their inputs and compute the estimate  $g = \frac{\Delta \cdot n}{s}$ . By the Chernoff bound,  $g$  is an  $(\epsilon, \delta)$ -approximation of  $d$ .

Note that the randomized function  $g(a, b)$  induced by the above sketching protocol is functionally private with respect to  $d_h(a, b)$ . We now show how to privately compute  $g$  with a small communication complexity. Our main tool is a private protocol for comparing a randomly sampled pair of bits. Formally, the protocol computes the randomized function **Sample-XOR**, defined as

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

Note that a private protocol for **Sample-XOR** must keep the choice of  $r$  secret from each party.

Figure 1 describes a private protocol, **Private-Sample-XOR**, for the function **Sample-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  $x_i \oplus m$ .



**Lemma 5.1** *Private-Sample-XOR is a private protocol computing the randomized function **Sample-XOR**.*

**Proof:** 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 parties output

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

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

Intuitively, 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  $b_{(r_A+r_B)}$ . The privacy can be formally argued by describing a simulator for each party. In fact, given the composition theorem (Theorem 2.2), it suffices to prove the privacy of an oracle-aided version of the protocol in which each of the two invocations of the  $\binom{n}{1}$ -OT protocol is replaced by a call to an oracle computing the corresponding function. A simulator for Alice's view in this oracle-aided protocol proceeds as follows. On input  $a \in \{0, 1\}^n$  and output value  $z \in \{0, 1\}$ :

1. Pick at random  $m_A \stackrel{\mathbb{R}}{\leftarrow} \{0, 1\}$ ,  $r_A \stackrel{\mathbb{R}}{\leftarrow} [n]$ , and  $z_A \stackrel{\mathbb{R}}{\leftarrow} \{0, 1\}$ .

2. Output  $(m_A, r_A)$  as Alice's random input,  $z_A$  as the output Alice receives from the first call to the  $\binom{n}{1}$ -OT oracle in Step 2, and  $z'_B \stackrel{\text{def}}{=} z \oplus z_A \oplus m_A$  as the message received from Bob in Step 3. (The remainder of Alice's view follows deterministically from her input and this simulated information.)

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

We argue that the view produced by the above simulator, conditioned on the inputs  $(a, b)$  and the output  $z$ , is distributed exactly as in the real (oracle-aided) protocol. This follows from the facts that: (1) Alice's random inputs  $m_A, r_A$  in the real protocol are independent of the inputs  $(a, b)$  and the output  $z$ , and are thus distributed in the simulated view as they should; (2) the output  $z_A$  received from the  $\binom{n}{1}$ -OT oracle in the real protocol is independent of  $a, b, m_A, r_A, z$ , as in the simulated view. ■

Given approximation parameters  $\epsilon$  and  $\delta$ , our private sampling estimator for the high distance case is implemented using  $s = O((n/d_{\min}) \cdot \log(1/\delta)/\epsilon^2)$  parallel invocations of Private-Sample-XOR. Its properties are summarized by the following lemma.

**Lemma 5.2 (Private approximation for the high distance case.)** *Let  $\mathcal{OT}$  be an arbitrary  $\binom{n}{1}$ -OT protocol (with security against a passive adversary). Then there exists a protocol  $\pi_{\text{high}}$  for approximating  $d_h(a, b)$  whose communication complexity is  $\tilde{O}((n/d_{\min}) \log(1/\delta)/\epsilon^2)$  times that of  $\mathcal{OT}$ , and whose round complexity is 1 plus that of  $\mathcal{OT}$ , such that:*

- If  $d = d_h(a, b) \geq d_{\min}$ , the protocol  $\pi_{\text{high}}$  outputs an  $\epsilon$ -approximation of  $d$  with overwhelming probability.
- The output  $g$  of  $\pi_{\text{high}}$  is functionally private with respect to  $d_h$ .
- $\pi_{\text{high}}$  privately computes its output. Specifically, Alice (resp., Bob) can simulate her view on input  $(a, b)$ , conditioned on an output  $g$ , based on  $g$  and her input  $a$  (resp.,  $g$  and his input  $b$ ) alone.

**Proof:** The protocol  $\pi_{\text{high}}$  proceeds as described above: the parties invoke Private-Sample-XOR  $s = O((n/d_{\min}) \cdot \log(1/\delta)/\epsilon^2)$  times in parallel, let  $\Delta$  be the sum of the  $s$  outputs, and output  $g = \Delta \cdot n/s$ . The approximation quality of the output  $g$  follows from a Chernoff bound, and its functional privacy follows from its symmetry.

Viewing the above protocol as an oracle-aided protocol (making oracle calls to the randomized Sample-XOR function), a simulator for Alice or Bob may proceed as follows: (1) let  $\Delta = gs/n$ ; (2) pick a random  $s$ -tuple  $(z_1, \dots, z_s) \in \{0, 1\}^s$  such that  $\sum z_i = \Delta$ ; (3) output  $(z_1, \dots, z_s)$  as the outputs of the  $s$  invocations of the Sample-XOR oracle. Again, this simulator perfectly emulates the oracle-aided protocol. This follows from the fact that the outputs of the Sample-XOR invocations in the real protocol, conditioned on the value of  $\Delta$ , are uniformly distributed subject to the restriction that their sum is  $\Delta$ . ■

Note that the sampling estimator does not give a reliable estimate when the distance  $d$  is significantly smaller than  $d_{\min}$  because its variance is too high. (That is, it is likely that no differences will be detected.)

## 5.2 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:

- (1) The induced function  $g$  is essentially determined by  $d_h$ . That is, except with negligible probability,  $g(a, b)$  takes a specific value determined by  $d_h(a, b)$ .
- (2) The above value is equal to  $d_h(a, b)$  if  $d_h(a, b) \leq d_{\max}$  and equal to “fail” otherwise.

Property (1) is used to implement a private computation of the low distance estimator. Property (2) guarantees functional privacy of the output over the entire range of inputs. It also guarantees correctness in case the distance is low. Indeed, for any sketching protocol satisfying property (1), a private computation of  $g$  (the randomized function induced by the sketching protocol) may proceed according to the naive approach described above. That is, the parties may locally compute the sketches based on their inputs and

a common random input, and then 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  $g$ . More formally, a simulator can pick  $r$  independently of the input and output, and then simulate the private protocol for  $G$  on the sketch induced by  $r$  and the original output. This simulation is correct because  $r$  is statistically close to uniform when conditioned on the input and output of  $g$ . Note, however, that 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.

We describe two different sketching protocols. The first is based on hashing, has a self-contained description, and only requires the private computation of a very simple reconstruction function.<sup>8</sup> We then give an alternate protocol based on Reed-Solomon codes. Its reconstruction function  $G$  is more complex, which makes the communication complexity of privately computing it higher.

### 5.2.1 A protocol based on hashing

Let the common random input define several independent (2-universal) hash functions. Given a correctness parameter  $k$  (where  $k = O(\log(1/\delta) \log d_{\max})$  is sufficient to guarantee an error bound  $\delta$ ), 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 of equal size. With probability  $1 - 2^{-\Omega(k)}$ , no bucket gets more than  $k \log d_{\max}$  bits in which  $a$  and  $b$  differ.
2. For each of the  $d_{\max}$  buckets, further partition its bits into  $(k \log d_{\max})^2$  sub-buckets. Now, if a given bucket contains at most  $k \log d_{\max}$  differences, then each of its sub-buckets contains 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  $1 \leq i \leq d_{\max}$ ,  $1 \leq j \leq (k \log d_{\max})^2$ ,  $1 \leq h \leq k$ ).
3. Hash the contents of each sub-bucket  $B_{ijh}$  to a  $k$ -bit string  $\beta_{ijh}$ .

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

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

**Lemma 5.3** *Suppose that  $d_h(a, b) \leq d_{\max}$ . Then, with probability  $1 - 2^{-\Omega(k)} \cdot d_{\max}$ ,*

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

**Proof:** 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. (2). 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 cannot be used. In this case, however, a very simple modification to the reconstruction function can fix this situation. The modified reconstruction first computes an estimate  $\tilde{d}$  by applying the right hand side of Eq. (2) to the sketches, and then it outputs  $\tilde{d}$  if  $\tilde{d} \leq d_{\max}$  and outputs “fail” otherwise.

The properties of the modified sketching protocol, denoted  $(S_{\text{hash}}, G_{\text{hash}})$ , are summarized in the following lemma.

---

<sup>8</sup>We note that this is hashing in the basic algorithmic sense; we do not require a cryptographic hash function.



**Lemma 5.4** *Letting  $k = \tilde{O}(1)$ , the sketching protocol  $(S_{\text{hash}}, G_{\text{hash}})$  and the induced randomized function  $g_{\text{hash}}$  satisfy the following properties:*

- *The output length of  $S_{\text{hash}}$  is  $\tilde{O}(d_{\text{max}})$ , and so is the circuit size of the reconstruction function  $G_{\text{hash}}$ .*
- *if  $d = d_h(a, b) \leq d_{\text{max}}$ , then  $g_{\text{hash}}(a, b) = d$  with overwhelming probability.*
- *if  $d > d_{\text{max}}$  then  $g_{\text{hash}}(a, b)$  outputs “fail” with overwhelming probability.*

**Proof:** The specified complexity bounds follow easily from the description of  $(G_{\text{hash}}, S_{\text{hash}})$ . In particular, the circuit size required for computing the right hand side of Eq. (2) is linear in the length of the sketch.

The first correctness property follows from Lemma 5.3. The second follows from the fact that when  $d > d_{\text{max}}$ , the right-hand side of Eq. (2) is bigger than  $d_{\text{max}}$  with overwhelming probability. This can be shown similarly to the proof of Lemma 5.3.  $\blacksquare$

### 5.2.2 A protocol based on Reed-Solomon Codes

We now describe an alternative to the sketching protocol  $(S_{\text{hash}}, G_{\text{hash}})$  of Section 5.2. This protocol satisfies all the properties of  $(S_{\text{hash}}, G_{\text{hash}})$  guaranteed by Lemma 5.4, except that the circuit size of its reconstruction function  $G$  is  $\tilde{O}(d_{\text{max}}^2)$  instead of  $\tilde{O}(d_{\text{max}})$ .

We start by describing a simpler variant of this protocol which does not give a reliable indication for its failure in the case where the distance is high. This variant relies on error-correcting codes for finding the locations in which two strings differ. A similar use of error-correcting codes in a related context was previously made in the communication complexity literature (see [21] and references therein).

Let  $F$  be a finite field with  $|F| > n$ . 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  $2d_{\text{max}} + 1$ , dimension  $n$ , and length  $n + 2d_{\text{max}}$ . The matrix  $H$  has  $2d_{\text{max}}$  rows and  $n$  columns. For any  $x \in F^n$  such that  $w_h(x) \leq d_{\text{max}}$ ,  $x$  can be uniquely recovered from the syndrome  $Hx$  (as  $x$  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  $d_{\text{max}}$ . 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  $d_h(a, b)$  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  $O(d_{\text{max}} \log n)$ . The circuit complexity of recovering the “errors” from the syndrome can be made as low as  $\tilde{O}(d_{\text{max}}^2)$  (see [19] and references therein) or even slightly sub-quadratic (using the polynomial factorization algorithm from [38]).

The output function  $g$  induced by the above sketching protocol does not reliably indicate failure when the distance is larger than  $d_{\text{max}}$ . This follows from the fact that there exist  $x, x'$  such that  $w_h(x) = w_h(x') > d_{\text{max}}$  and yet applying the decoding procedure to  $Hx$  and  $Hx'$  yields a different number of errors. We now modify the above construction such that if  $d > d_{\text{max}}$  it outputs “fail” with overwhelming probability.

The modified sketching protocol uses a  $k$ -bit random input  $r$ , where  $r$  is interpreted as a key to a pseudorandom function  $h_r : [n] \rightarrow \text{GF}(2)^k$ , and where  $k = \tilde{O}(1)$ . The  $n$  possible outputs of  $h_r$  define a pseudorandom  $k \times n$  matrix  $R$  over  $\text{GF}(2)$ , satisfying the following properties: (1) the  $i$ th column of  $R$  can be computed from  $r$  by a circuit of size  $\tilde{O}(k) = \tilde{O}(1)$ ; (2) for any nonzero  $x \in \text{GF}(2)^n$ , 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 pseudorandom functions for simplicity; more efficient constructions can be based on small-bias probability spaces [49].) 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 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  $w_h(v_e) \leq d_{\text{max}}$ , and that  $v_e = (a - b)$  if and only if  $d_h(a, b) \leq d_{\text{max}}$ . 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  $\tilde{O}(k^{O(1)} \cdot d_{\text{max}}^2) = \tilde{O}(d_{\text{max}}^2)$ ; (2) if  $d = d_h(a, b) \leq d_{\text{max}}$ ,  $g(a, b) = d$  with probability 1; (3) if  $d > d_{\text{max}}$ , then  $g(a, b)$  outputs “fail” with

overwhelming probability. Our final sketching protocol thus satisfies all the desired properties guaranteed by Lemma 5.4, up to a quadratic blowup in the cost of reconstruction.

**Remark: (Application to communication complexity).** Our sketching methods for the low distance case can be applied to solve the following communication complexity problem posed by Yao [56]. Suppose that Alice and Bob each hold an input string of length  $n$  as well as a common random input. They wish to determine whether the Hamming distance between their inputs is bounded by  $d$ . To this end, they each send a message to a referee, who should output the correct answer with high probability (say greater than  $2/3$ ). Our sketching methods for the low distance case directly yield solutions to this problem. The first method gives a protocol whose communication complexity is  $O(d \cdot \text{polylog } d)$ , whereas the method based on Reed-Solomon codes gives a protocol whose communication complexity is  $O(d \log n)$ .<sup>9</sup> We note that the dependence of the latter bound on  $\log n$  is inherent to the coding-based approach, as the sketch reveals not only the number of places where the two inputs differ but also their locations. In contrast, the hashing-based approach reveals only the Hamming distance between the inputs. Finally, a similar complexity can also be obtained in the standard two-party communication complexity model via a suitable derandomization of the common random string, either under cryptographic assumptions (using a general-purpose pseudorandom generator) or unconditionally (using limited independence).

### 5.2.3 Using the protocols

Based either on the hashing-based sketching protocol or the Reed-Solomon-based sketching protocol, a private protocol for the low distance case may be constructed as outlined in the beginning of this section. In the following, we restrict our attention to the hashing-based variant because of its better efficiency and simplicity.

**Lemma 5.5 (Private approximation for the low distance case.)** *Suppose any of the assumptions of Theorem 2.3 holds. Then, for any  $1 \leq d_{\max}(n) \leq n$ , there exists a three-round protocol  $\pi_{\text{low}}$  with  $\tilde{O}(d_{\max})$  communication, such that:*

- If  $d = d_h(a, b) \leq d_{\max}$ , the protocol  $\pi_{\text{low}}$  outputs the exact value of  $d$  with overwhelming probability;
- If  $d = d_h(a, b) > d_{\max}$ , the protocol  $\pi_{\text{low}}$  outputs “fail” with overwhelming probability.
- The output  $g$  of  $\pi_{\text{low}}$  is statistically indistinguishable from some function  $g'$  that is functionally private with respect to  $d_h$ .
- $\pi_{\text{low}}$  privately computes its output.

**Proof:** Let  $(S, G)$  be any sketching protocol satisfying the properties of Lemma 5.4. The required protocol  $\pi_{\text{low}}$  proceeds as follows. In the first round, Alice sends to Bob a seed (of length  $\tilde{O}(1)$ ) to a pseudorandom generator which is used to produce a sufficiently long common random input. Then, each party locally applies the sketching function  $S$  to its input and the common random input, and together they invoke a protocol for privately evaluating the reconstruction function  $G$  on their sketches. Using Theorem 2.3, this requires  $\tilde{O}(d_{\max})$  communication and either two additional rounds, assuming homomorphic encryption, or three rounds, assuming trapdoor permutations. (In the variant based on trapdoor permutations, the first round does not depend on the inputs and can thus be done in parallel to the first message from Alice to Bob. Thus, the main protocol can be implemented in three rounds under either type of assumption.)

We now argue that  $\pi_{\text{low}}$  satisfies the four required properties. The first two follow immediately from the assumptions on  $(S, G)$  and from the properties of a pseudorandom generator. The functional privacy property follows by defining  $g'(a, b)$  as  $d_h(a, b)$  if this distance is at most  $d_{\max}$  and “fail” otherwise. Finally, the following simulator shows the privacy of  $\pi_{\text{low}}$ . As before, we describe Alice’s simulator of the oracle-aided version of the protocol having oracle access to  $G$ ; Bob’s simulator is similar. On input  $a$  and  $g$ :

- Pick a random seed  $\alpha$  to the pseudorandom generator.

<sup>9</sup>The method based on Reed-Solomon codes was independently used in a similar context by Gavinsky et al. [25].

- Output  $\alpha$  as Alice’s random input and  $g$  as the output of the  $G$ -oracle. (The message sent by Alice to the  $G$ -oracle is determined by  $a$  and  $\alpha$ .)

The correctness of the above simulator follows from the fact that all but a negligible fraction of the possible seeds  $\alpha$  would lead to the same output  $g$ . Thus, in the real-life execution of  $\pi_{\text{low}}$ , the distribution of the seed  $\alpha$  conditioned on the inputs  $(a, b)$  and the output  $g$  is statistically close to uniform. ■

### 5.3 The Combined Protocol

Using the protocols  $\pi_{\text{low}}$  and  $\pi_{\text{high}}$  of Lemma 5.2 and Lemma 5.5 as subprotocols, our full protocol  $\pi_h$  proceeds as follows. Given the desired approximation quality  $\epsilon$ :

- Invoke protocol  $\pi_{\text{high}}$  of Lemma 5.2 with parameters  $\epsilon$  and  $d_{\text{min}} = n^{1/2}/\epsilon$ . Let  $d_1$  denote its output.
- In parallel, invoke protocol  $\pi_{\text{low}}$  of Lemma 5.5 with parameter  $d_{\text{max}} = n^{1/2}/\epsilon$ . Let  $d_2$  denote its output.
- If  $d_2 = \text{“fail”}$ , output  $d_1$ ; else output  $d_2$ .

**Lemma 5.6** *The above protocol  $\pi_h$  is a private  $\epsilon$ -approximation protocol for  $d_h$  in the liberal sense.<sup>10</sup>*

**Proof:** The randomized function  $\hat{d}$  computed by  $\pi_h$  is obtained from the outputs  $d_1$  and  $d_2$  of  $\pi_{\text{high}}$  and  $\pi_{\text{low}}$ , respectively. By the functional privacy properties of  $d_1$  and  $d_2$  with respect to  $d_h$  (see Lemmas 5.2 and 5.5), the final output  $\hat{d}$  is also (indistinguishable from being) functionally private with respect to  $d_h$ .

The  $\epsilon$ -approximation property of  $\hat{d}$  follows from the facts that: (1) if  $d > d_{\text{max}}$  then (with overwhelming probability) the final output  $\hat{d}$  is produced by  $\pi_{\text{high}}$ , and, because  $d > d_{\text{max}} \geq d_{\text{min}}$ , this output is  $\epsilon$ -correct; (2) if  $d \leq d_{\text{max}}$ , then the low distance subprotocol produces  $\hat{d}$ , which is guaranteed in this case to be correct with overwhelming probability.

It remains to show that  $\pi_h$  satisfies the liberal privacy requirement of Definition 7. As usual, we describe a simulator for Alice, replacing  $\pi_{\text{high}}$  and  $\pi_{\text{low}}$  by oracle calls to the functions computed by these protocols. This is justified by Lemmas 5.2 and 5.5 and Theorem 2.2.

On inputs  $a$ ,  $d = d_h(a, b)$ , and  $\hat{d}$ , the simulator proceeds as follows:

- Sample  $d_1$  conditioned on  $d$  and  $\hat{d}$ . That is, if  $d > d_{\text{max}}$ , let  $d_1 = \hat{d}$ ; otherwise, sample  $d_1$  from a binomial distribution with parameters  $s$  and  $d/n$  (where the number of trials  $s$  is as in  $\pi_{\text{high}}$ ), then multiply by  $n/s$ .
- Compute  $d_2$  from  $d$ . That is, let  $d_2 = d$  if  $d \leq d_{\text{max}}$  and  $d_2 = \text{“fail”}$  otherwise.
- Output  $d_1$  as the output of the oracle corresponding to  $\pi_{\text{high}}$  and  $d_2$  as the output of the oracle corresponding to  $\pi_{\text{low}}$ .

The correctness of the above simulator follows from the fact that the joint distribution of  $(\hat{d}, d_1, d_2)$  induced by the simulator is statistically indistinguishable from that of the real-life protocol. ■

**Remark: (On strict vs. relaxed privacy).** As described,  $\pi_h$  does not satisfy the stricter notion of private approximation defined in Definition 6. Indeed, the intermediate outputs  $d_1$  and  $d_2$  may give more information than is implied by the protocol’s final output  $\hat{d}$ . Specifically, when the output  $\hat{d}$  is slightly lower than the threshold  $d_{\text{max}}$ , the output alone does not determine whether  $d > d_{\text{max}}$ , whereas the output  $d_2$  of  $\pi_{\text{low}}$  does. Thus,  $\pi_h$  does not privately compute its output  $\hat{d}$ . However,  $\pi_h$  can be easily modified to satisfy the stricter privacy requirement. One way of achieving this is by hiding all intermediate results except the final outcome. This can be done by modifying  $\pi_{\text{high}}$  and  $\pi_{\text{low}}$  so that their outputs are “secret-shared” between the parties and applying another private protocol to compute  $\hat{d}$  from the shared outputs of the subprotocols. A more efficient alternative is to incorporate the additional information revealed by  $\pi_h$  into

<sup>10</sup>This refers to the relaxed notion of private approximation defined in Definition 7. Modifications of  $\pi_h$  that satisfy the strict definition are discussed in the remark following the proof of this lemma.

its output  $\hat{d}$ . This can be achieved by slightly perturbing the output, so that its value encodes  $d_1, d_2$  without significantly changing the approximation quality.

Substituting the complexity parameters of the two subprotocols and the  $\binom{n}{1}$ -OT protocol of Theorem 2.4 yields the main result of this section:

**Theorem 5.7** *Assuming the existence of homomorphic encryption, the Hamming distance function can be privately  $\epsilon$ -approximated with communication complexity  $\tilde{O}(n^{1/2}/\epsilon)$  and three rounds of interaction.*

In Section 5.4, we show that it is possible to obtain improved efficiency in a relaxed model with offline communication.

**Remark: (On computational complexity).** A naive implementation of the combined protocol  $\pi_h$  (corresponding to Theorem 5.7) has computational complexity of  $\tilde{O}(n^{3/2}/\epsilon)$ . This complexity is dominated by  $\tilde{O}(n^{1/2}/\epsilon)$  invocations of the  $\binom{n}{1}$ -OT primitive, each requiring  $\tilde{O}(n)$  time. However, the cost of these invocations can be amortized [34], yielding a protocol with the same asymptotic communication complexity and (an essentially optimal) computational complexity of  $\tilde{O}(n/\epsilon)$ .

**Remark: (On using Reed-Solomon codes).** Recall that the reconstruction function of our sketching protocol based on Reed-Solomon codes is less efficient than that of our hashing-based sketching protocol. Nonetheless, it can still be used to give a private approximation protocol for the Hamming distance with sublinear communication complexity. Specifically, setting  $d_{\min}$  and  $d_{\max}$  in the combined protocol  $\pi_h$  to  $\tilde{O}(n^{1/3})$  and using Reed-Solomon-based sketching in  $\pi_{\text{low}}$ , the resulting communication complexity is  $\tilde{O}(n^{2/3})$ .

## 5.4 Polylogarithmic $L^2$ Protocol 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  $(\sum |a_i - b_i|^2)^{1/2}$ , where  $\langle a_i \rangle$  and  $\langle b_i \rangle$  are sequences of integers.<sup>11</sup> A solution for the Hamming distance follows as a special case. Essentially, we verify that the protocol from [32] 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.<sup>12</sup> These samples are encrypted using *homomorphic public-key encryption*<sup>13</sup>—i.e., anyone can form an encryption  $E(\alpha, \kappa)$  of  $\alpha$  that can be decrypted only by knowing the secret key  $\kappa$ , and, from encryptions  $E(\alpha, \kappa)$  and  $E(\beta, \kappa)$  of  $\alpha$  and  $\beta$  for the same secret key  $\kappa$ , anyone can form an encryption  $E(\alpha + \beta, \kappa)$  of  $\alpha + \beta$  for  $\kappa$ . Using a threshold homomorphic encryption scheme, Alice and Bob split  $\kappa$  so that neither can decrypt alone but together they can decrypt.

As prescribed in [32], Alice should form  $\sum_i a_i s_i$ . In our context, she forms  $E(\sum_i a_i s_i, \kappa)$ , 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  $E(\sum_i a_i s_i, \kappa)$ , using the homomorphic properties of the encryption. Alice and Bob then form  $E(\sum_i s_i (a_i - b_i), \kappa)$ , again using the homomorphic properties of the encryption. The insecure protocol prescribes that they compute  $(\sum_i s_i (a_i - b_i))^2$ , repeat, and take the median of means, using Lemma 2.1. In our setting, Alice and Bob perform the median of means of squares of decryptions of  $E(\sum_i s_i (a_i - b_i), \kappa)$ -values using a secure multiparty computation. (This can be described with 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. Privacy of the messages is immediate by construction.

<sup>11</sup>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.

<sup>12</sup>Actually, the samples are indistinguishable from finite-precision approximations to real-valued Gaussian samples. This suffices; see Section 4.

<sup>13</sup>As is the case throughout this paper, we assume that an adversary with resources polynomial in  $n$  cannot break the encryption. In this section, however, we need to assume that cryptographic operations such as decryption (with the key) and homomorphic transformation can be done in time polylogarithmic in  $n$ , i.e., in time comparable to the time needed for other operations. Similarly, we assume that a ciphertext is longer than a cleartext by at most a factor polylogarithmic in  $n$ . That is, we assume exponential-strength cryptographic operations. If only weaker cryptographic operations are available, the cryptographic operations become the efficiency bottleneck.

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  $\|\langle a_i \rangle - \langle b_i \rangle\|_2$ —i.e., 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.

## 6 Secure Approximations of #P-hard Functions

We now turn our attention to securely approximating natural #P-hard problems, where the goal is to achieve polynomial-time secure approximations. This is in contrast to problems on massive data sets that we have been focusing on thus far, where polynomial time exact computation is possible, and the goal is to achieve lower complexity (sublinear in the Hamming distance case). Thus, throughout this section, “efficient” should be interpreted as “probabilistic polynomial time”. By Theorem 3.1, in the current setting, it is sufficient to design any efficiently computable private approximation for the problem at hand.

We start by observing that artificially constructing #P-hard problems which satisfy the above property 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  $\epsilon$ ,  $2^{2n}$  is a  $(1 \pm \epsilon)$ -factor private approximation to  $g(x)$ , this approximation does not approximate any interesting quantity. Thus, in general, while some exact #P-hard problem may be interesting, their approximate versions may not be.

In this section, we give private approximations to natural #P-hard problems, most notably the permanent, the most well known #P-complete problem (Section 6.1). We also discuss extensions of our methods to other #P hard problems (Section 6.2).

### 6.1 Secure Approximation of the Permanent

The *permanent* of a matrix  $M$  is defined as  $\text{per}(M) = \sum_{\pi} \prod_{i=1}^n M(i, \pi(i))$ , where all permutations  $\pi$  on  $\{1, \dots, n\}$  contribute to the sum. For a 0/1-valued matrix  $M$ ,  $\text{per}(M)$  counts the number of perfect matchings in the corresponding bipartite graph (defined by the adjacency matrix  $M$ ).

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*—namely, any approximation to the permanent yields (a polynomially related) approximation to these problems. Clearly, a private approximation to the permanent immediately yields a private approximation to any problem that reduces to the permanent in an approximation-preserving way. 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. As another example, the Pauling bond order of an edge, in a certain graph representation of a molecule, reduces to a permanent computation in an approximation-preserving way. We omit the definition of the Pauling bond order here, but note that it serves as a useful theoretical prediction of the physical strength of a molecular bond.

In this section, we show how to privately compute an approximation of the permanent of a shared matrix in polynomial time. Specifically, let  $f(M_1, M_2) = \text{per}(M_1 + M_2)$ , where  $M_1$  and  $M_2$  are  $n \times n$  matrices with  $n$ -bit non-negative entries.<sup>14</sup> By Theorem 3.1, it is sufficient to obtain an efficiently computable functionally private approximation for the permanent.

#### Non-Secure FPTAS for the Permanent

A string of results [37, 10, 35], culminating in the recent result of Jerrum, Sinclair, and Vigoda [36], provides efficient approximation algorithms for the permanent of an arbitrary matrix with non-negative entries. We build on their techniques to construct our functionally private approximation. For our purposes, the algorithm of [36] may be viewed as consisting of the following stages:

<sup>14</sup>We are slightly deviating from our previous notation in which  $n$  is the total length of each input.

1. Design an efficient randomized algorithm  $A$  satisfying the following requirement. For any  $n \times n$  matrix  $M$  and  $1 \leq i \leq n$ , the output of  $A(M, i)$  is Bernoulli random variable with success probability negligibly far from  $p_i$  such that:
  - $\prod_i 1/p_i = \text{per}(M)$ ;
  - for all  $i$ ,  $p_i \geq 1/\text{poly}(n)$ .
2. Use sampling to approximate each  $p_i$  efficiently.
3. Multiply these approximations to obtain an approximation of the product (and thus of  $1/\text{per}(M)$ ).
4. Invert to get an approximation of  $\text{per}(M)$ .

We note that the central technical component of [36] (and its predecessors) is the construction of the random variables in stage 1, which uses a Monte Carlo Markov chain method to sample from the set of all perfect matchings on a graph with a distribution that is statistically indistinguishable from uniform.

### Achieving Functional Privacy

Our goal is to obtain a functionally private approximation to the permanent. Considering the algorithm of [36] outlined above, we first observe that the sampling-based approximation of each  $p_i$  given by stage 2 is already functionally private with respect to  $p_i$ . However, the product of approximations does potentially leak information about its factors (e.g., the standard deviation depends on the factors), and, thus, stage 4 results in a non-private output.

To avoid this leakage, one might be tempted to estimate the product at once; that is, if  $X_i$  is a Bernoulli random variable with success probability  $p_i$ , one might be tempted to estimate the expectation of  $Y = \prod X_i$  by sampling  $Y$ . This indeed results in an estimator with the right distribution perfectly simulatable from the product  $\prod p_i$ . However, approximating the product in this way is not efficient, as the product may be exponentially small (thus using only a polynomial number of samples, the produced “approximation” is likely to be zero).

Our goal then, reduces to designing an efficient, functionally private approximation for the product  $\prod 1/p_i$ , given coins with biases  $p_i$  as above. The bulk of the technical work involves designing, for each  $i$ , a coin with bias negligibly far from  $p_i^{1/n}$ . This is done by manipulating biases of coins to produce coins with new biases. We first describe some elementary manipulations and then combine them to construct the  $n$ th root. Finally, in Theorem 6.1, we show how to use the  $n$ th root construction.

Given success probabilities  $q_0, q_1$  and  $q_2$ , let  $A_i$  denote an event with probability  $q_i$  for  $i = 0, 1, 2$ . One can form an event with success probability  $q_0 q_1$  by taking the joint event  $A_0 A_1 = A_0 \wedge A_1$  of  $A_0$  and  $A_1$ , one can form an event with success probability  $1 - q_0$  by taking the complementary event  $\overline{A_0}$  of  $A_0$ , and one can form an event with success probability  $q_0 q_1 + (1 - q_0) q_2$ —a convex combination—by taking event  $A_1$  if  $A_0$  holds and taking event  $A_2$  if  $A_0$  fails, an event that we denote by  $(A_0 ? A_1 : A_2)$ . In this way, we have implemented functions on the real numbers  $q_0, q_1, q_2$  by manipulating events algorithmically, without the need to know  $q_0, q_1, q_2$ . We refer to the functions  $x \mapsto 1 - x$ ,  $(x, y) \mapsto xy$ , and  $(x, y, z) \mapsto xy + (1 - x)z$  as *elementary manipulations of biases*.

We now use these elementary manipulations to construct an event with success probability negligibly far from  $p^{1/n}$ , given a coin with (unknown) bias  $p$ , which is used to generate independent events with success probability  $p$ . We use the Taylor series for  $x^{1/n}$  at  $x = 1$ , which is  $\sum_j (-1)^j \binom{1/n}{j} (1 - x)^j$ . The coefficient of  $(1 - x)^j$  is

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

which is negative and is at most  $1/(nj)$  in absolute value. Thus, the sum of the absolute values of all but the leading coefficient in a  $O(nk)$ -term Taylor polynomial  $T(x)$  is at most  $\sum_{j=1}^{kn} \frac{1}{nj} \leq \frac{\log(kn)}{n}$ , which we can assume is less than 1. (Otherwise,  $k > 2^n/n$ , and we can solve the permanent exactly in time polynomial in  $k$ , by [48].) Thus  $T(x)$  is 1 less a sub-convex combination of  $(1-x), (1-x)^2, \dots, (1-x)^{kn}$ . We now make two important observations. First, for  $x \geq 1/(2n)$ , an  $O(nk)$ -term Taylor expansion for  $x^{1/n}$  around  $x = 1$  has error bounded by  $2^{-2k} = O(2^{-k}/n)$ . So  $O(nk)$  terms suffice for our purposes. Second, an  $\ell$ -term approximation to  $T(x)$  can be written as a circuit of gates of the form  $x \mapsto 1-x$ ,  $(x_1, x_2) \mapsto x_1x_2$ , and  $(x, y) \mapsto rx + (1-r)y$ ; all of these gates compute functions that are implementable as elementary manipulations of biases, as desired. It follows that, given a coin with bias negligibly far from unknown  $p$ , one can construct an experiment with success probability negligibly far from  $p^{1/n}$ , using tosses of the original coin. From the form of the expression for  $T(x)$ , it follows that we need at most  $\text{poly}(nk)$  tosses of the original coin and at most  $\text{poly}(nk)$  computation.

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

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

Isolating the leading 1 and using convex combinations instead of sums, we get

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

Suppose event  $A$  has unknown probability  $p$  and  $F_t$  (a coin flip) has success probability  $t$ . Then the following event, which can be constructed directly from the above expression for  $T(x)$ , has probability  $T(p)$ , where all occurrences in the formula of  $A$ 's and  $F$ 's are independent.

$$E = \overline{(F_{1/2} ? \bar{A} : \bar{A}^2 F_{1/4})}. \quad (3)$$

For a polynomial of degree  $\ell$ , the appropriate generalization of Event (3) uses just  $O(\ell^2)$  experiments of type  $A$  (i.e., at most  $O(\ell^2)$  random variables, all independent and distributed identically to the indicator random variable for  $A$ ) and constantly many  $F$  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  $(1 \pm 2^{-k})$ . 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 product construction.

Finally, we show how to use the new  $p \mapsto p^{1/n}$  construction. Because  $\prod p_i^{1/n} = 1/\text{per}(M)^{1/n} \geq 1/\text{poly}(n)$ , this product can be efficiently approximated by sampling directly from the joint distribution of coins with biases  $p_1^{1/n}, \dots, p_n^{1/n}$ , and, by raising to the power  $-n$ , we are done. This yields the following theorem.

**Theorem 6.1** *Let  $f(M_1, M_2) = \text{per}(M_1 + M_2)$ , where  $M_1, M_2$  are  $n \times n$  matrices with  $n$ -bit non-negative entries. Then, for any  $\epsilon(n) \geq 1/\text{poly}(n)$ , there is a polynomial-time private  $\epsilon(n)$ -approximation for  $f$ .*

**Proof:** Consider the overall algorithm from [36] described above, in which we want to compute  $\prod 1/p_i$  and, for each  $i$ , we have a coin with bias within the factor  $(1 \pm 2^{-k}/n)$  of  $p_i$ . Now consider the construction described above, in which we construct an event  $A_i$  with probability  $p_i'$  within the factor  $(1 \pm 2^{-k}/n)$  of  $p_i^{1/n}$ , given a coin with bias  $p_i$ . Then the joint event  $\bigwedge_i A_i$  has probability  $(\prod p_i)^{1/n} (1 \pm O(2^{-k}))$ . Because each  $p_i$  is at least  $1/\text{poly}(n)$  in [36], so is their geometric mean, so that  $\Pr(\bigwedge_i A_i) \geq 1/\text{poly}(n)$ . Using Lemma 2.1, estimate  $\Pr(\bigwedge_i A_i)$  from  $\text{poly}(n \log(1/\delta)/\epsilon)$  samples, getting  $\text{per}(M_1 + M_2)^{-1/n} (1 \pm O(\epsilon/n))$  with probability at least  $1 - \delta$ , and then take the  $-n$  power. By the preceding discussion, the result is efficient, approximately correct, and private.  $\blacksquare$

## 6.2 Extensions to Other #P-complete Problems

As discussed at the start of Section 6.1, secure approximation of the permanent immediately implies secure approximation for the large array of problems that reduce to the permanent in an approximation-preserving manner, some examples of which were presented. We now turn to showing how to generalize the techniques we used in the permanent approximation to work for a more general class of problems.

## General Secure Approximations Based on Monte-Carlo Methods

Our proof for the permanent built on a (non-secure) approximation based on Monte Carlo Markov chains. We now want to extend our techniques to work for other intractable functions  $f(a, b)$  that have polynomial-time approximation schemes based on a similar Monte Carlo Markov chain approach. Indeed, the technique of rapidly mixing Markov chains is inherently suited for use in functionally private approximations, as by the 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 depends 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 do not want to estimate the fraction of objects satisfying some property, but rather some function of several such fractions (such as the product). To this end, our techniques of manipulating probabilities and using  $j$ th roots (through a Taylor expansion estimation) are useful, as described in more detail below.

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.” The success probability  $3/4$  can be boosted up to  $1 - \delta$  by performing  $O(\log(1/\delta))$  repetitions.

We begin with a definition that intuitively says that  $\psi$  is an approximation-preserving function.

**Definition 9** *A deterministic real function  $\psi$  is polynomially relatively continuous if, for all  $x$  and for all  $\epsilon > 0$ , there exists  $\eta > 1/\text{poly}$  such that  $\psi(x \cdot (1 \pm \eta)) \subseteq \psi(x) \cdot (1 \pm \epsilon)$ .*

**Lemma 6.2** *Let  $\psi$  be a polynomially relatively continuous function that is easy to compute and to invert. Suppose  $f(a, b) = \psi(\Pr(\mathcal{E}))$ ,  $\Pr(\mathcal{E}) \geq 1/\text{poly}$ , where  $\mathcal{E}$  is an event (parameterized by  $a$  and  $b$ ) under a probability distribution,  $D$ , such that one can sample in polynomial time from a distribution that is statistically (respectively, computationally) indistinguishable from  $D$ . Then  $f(a, b)$  has a statistically (respectively, computationally) 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 2.1 and then apply  $\psi$ . To see that this is functionally private, note that, from  $f(a, b)$  alone, a sampling algorithm  $\mathcal{S}$  can construct an  $\Omega(k)$ -bit approximation to  $\Pr(\mathcal{E}) = \psi^{-1}(f(a, b))$ . It can then apply Lemma 2.1 to a Bernoulli random variable with success probability negligibly far from  $\Pr(\mathcal{E})$  and apply  $\psi$ . The result follows. ■

Before proceeding, we consider another transformation, not needed for the permanent:

**Lemma 6.3** *Fix known  $r \geq 1$  and small  $\tau_1, \tau_2 > 0$ . Suppose we can make independent tosses of a coin with unknown bias  $p$ , where  $p \geq \tau_1 \geq 1/\text{poly}$ . Suppose further that  $rp$  is known to be at most  $1 - \tau_2 \leq 1 - 1/\text{poly}$ . Then we can construct a coin with bias indistinguishable from  $rp$ .*

**Proof:** Suppose we are given a number  $r$  and a coin with bias  $p$ , bounded as above. We “enrich” the coin by the factor  $r$  when we do the following experiment: Toss the original coin  $N \approx (k + \ln(1/\tau_1))r^2/\tau_2^2$  times and let  $S$  denote the number of heads obtained. Toss one more coin, with bias  $\min(1, rS/N)$ , and output the result of the last coin. Let  $p'$  denote the overall probability of success. We now show that  $p' = rp(1 \pm 2^{-k})$ .

Write the probability that the constructed coin succeeds as

$$p' = \sum_s \Pr(1|S = s) \Pr(S = s) = \sum_s \min\left(1, \frac{rs}{N}\right) \Pr(S = s).$$

One direction is easy—namely,

$$\begin{aligned} p' &= \sum_s \min\left(1, \frac{rs}{N}\right) \Pr(S = s) \\ &\leq \sum_s \frac{rs}{N} \Pr(S = s) \\ &= \frac{r}{N} E[S] \\ &= rp, \end{aligned}$$



as desired. Thus we need to show that  $p' \geq rp(1 - 2^{-k})$ , which we do by bounding the probability that  $\min(1, \frac{rS}{N}) = 1$ .

Note that  $rS/N > 1$  iff  $S$  exceeds its mean of  $pN$  by at least  $(1/r - p)N$ . By the Chernoff inequality, because  $(1/r - p) = (1 - rp)/r \geq \tau_2/r$ , this occurs with probability at most  $e^{-\Theta((1/r-p)^2 N)} \leq e^{-\Theta(\tau_2^2 N/r^2)}$ , and, below, we want this to be less than  $\tau_1 2^{-k}$ . For that, it suffices that  $N = \Theta(k + \log(1/\tau_1))r^2/\tau_2^2$ .

Next, observe that if  $B_{p'}$  is a Bernoulli random variable with success probability  $p'$ , then

$$\begin{aligned}
p' = E[B_{p'}] &= \sum_s \min(1, rs/N) \Pr(S = s) \\
&= \sum_s (rs/N) \Pr(S = s) - \sum_{rs/N > 1} (rs/N - 1) \Pr(S = s) \\
&\geq \sum_s (rs/N) \Pr(S = s) - \sum_{rs/N > 1} (rs/N) \Pr(S = s) \\
&\geq r \left( \sum_s (s/N) \Pr(S = s) - \sum_{rs/N > 1} (s/N) \Pr(S = s) \right) \\
&\geq r \left( \sum_s (s/N) \Pr(S = s) - \sum_{rs/N > 1} \Pr(S = s) \right) \\
&\geq r(p - \Pr(rs/N > 1)) \\
&\geq r(p - \tau_1/2^k) \\
&\geq rp(1 - 2^{-k}).
\end{aligned}$$

■

We now return to general Monte Carlo Markov chain methods. 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.

**Theorem 6.4** *Let  $\psi$  be a polynomially relatively continuous function that is easy to compute and to invert. Suppose  $f(a, b) = \psi(\phi(\Pr(\mathcal{E}_1), \Pr(\mathcal{E}_2), \dots, \Pr(\mathcal{E}_j)))$ , where each event has probability at least  $1/\text{poly}$  in a probability distribution that can be nearly sampled in polynomial time, and where  $\phi$  is a polynomial-sized, constant-depth arithmetic formula with gates of the following form:*

- $t \rightarrow 1 - t$
- $t_1, t_2 \rightarrow t_1 t_2$
- $\perp \rightarrow r$ , where  $r \in [1/\text{poly}, 1 - 1/\text{poly}]$  (Here  $\perp$  denotes the empty input. The number  $r$  must be efficiently constructible; e.g., the  $\ell$ th bit of  $r$  should be computable in time polynomial in  $\ell$ .)
- $(t_1, t_2, \dots, t_\ell) \rightarrow \sum_i r_i t_i$ , where  $\sum_i r_i = 1$
- $t \rightarrow t^r$ , for  $1/\text{poly} \leq r \leq 1$
- $t \rightarrow rt$ , for  $r \geq 1$ , under the promise that  $1/\text{poly} < t$  and  $rt < 1 - 1/\text{poly}$
- $(t_1, t_2, \dots, t_\ell) \rightarrow \prod_i t_i^{r_i}$ , where  $\sum_i r_i = 1$  and each  $r_i > 1/\text{poly}$ .

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

**Proof:** We show that each gate in  $\phi$  satisfies the following invariant: If each input takes values in  $[1/\text{poly}, 1 - 1/\text{poly}]$ , each input can be approximated in polynomial time by sampling, and, for each input, there is a polynomial-time-constructible Bernoulli experiment with success probability negligibly far from the ideal value, then the output satisfies the same three conditions:

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

The first conclusion is clear for each of the gates. The second conclusion follows from the first and third conclusions, the hypothesis about estimation of events  $\mathcal{E}_i$  by sampling, and Lemma 2.1. As for the third conclusion, we consider the allowed types of gates in turn. We show, for each gate  $g$ , that we can construct a coin with bias differing negligibly from the output value of  $g$ , given coins with biases equal to the input values to  $g$ , such that the total number of coins required by  $g$  is polynomial. Each gate above was discussed in Section 6.1 or in Lemma 6.3.

As in Lemma 6.2, it follows that we can estimate  $f$  by estimating  $\phi$  and then applying  $\psi$ . Also as in Lemma 6.2, to see that this approximation is functionally private, from  $f(a, b)$ , a sampling algorithm  $\mathcal{S}$  can compute

$$\psi^{-1}(f(a, b)) = \phi(\Pr(\mathcal{E}_1), \Pr(\mathcal{E}_2), \dots, \Pr(\mathcal{E}_j)),$$

apply Lemma 2.1 to a Bernoulli random variable with success probability indistinguishable from  $\psi^{-1}(f(a, b))$ , then apply  $\psi$ . The result follows. ■

## Acknowledgments

We thank the anonymous TALG referees for many helpful comments and suggestions. We thank Dana Randall for suggesting applications of the permanent described in Section 6.1 and Jessica Fong for helpful discussions and collaboration in early stages of this work. Finally, we are grateful to Adam Smith for helpful discussions and pointers concerning the complexity of decoding Reed-Solomon codes from their syndrome.

## References

- [1] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proc. ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, 2000.
- [2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002. Earlier version in PODS ’99.
- [3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. Earlier version in STOC ’96.
- [4] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 1992.
- [5] Z. Bar-Yossef. Personal communication, 2004.
- [6] D. Beaver. Foundations of secure interactive computing. In *Advances in Cryptology — CRYPTO ’91*, LNCS 576, pages 377–391. Springer-Verlag, 1991.
- [7] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. 22th Annual ACM Symposium on the Theory of Computing*, pages 503–513, 1990.
- [8] A. Beimel, P. Carmi, K. Nissim, and E. Weinreb. Private approximation of search problems. In *Proc. 38th Annual ACM Symposium on the Theory of Computing*, pages 119–128, 2006.
- [9] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th Annual ACM Symposium on the Theory of Computing*, pages 1–10. ACM Press, 1988.
- [10] A. Broder. How hard is it to marry at random? In *Proc. 18th Annual ACM Symposium on the Theory of Computing*, pages 50–58, 1986. Erratum in 20th STOC, p. 551.

- [11] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology — EUROCRYPT '99*, LNCS 1592, pages 404–414. Springer-Verlag, 1999.
- [12] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.
- [13] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.
- [14] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright. Selective private function evaluation with applications to private statistics. In *Proc. 20th Annual ACM Symposium on Principles of Distributed Computing*, pages 293–304. ACM Press, 2001.
- [15] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th Annual ACM Symposium on the Theory of Computing*, pages 11–19, 1988.
- [16] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45:965–981, 1998. Earlier version in FOCS '95.
- [17] G. Cormode, M. Paterson, S. Sahinalp, and U. Vishkin. Communication complexity of document exchange. In *11th Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 197–206, 2000.
- [18] DIMACS special year on massive data sets, 1997–1999. [http://dimacs.rutgers.edu/SpecialYears/1997\\_1998/](http://dimacs.rutgers.edu/SpecialYears/1997_1998/).
- [19] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology — EUROCRYPT '04*, LNCS 3027, pages 523–540. Springer-Verlag, 2004.
- [20] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28:637–647, 1985.
- [21] T. Feder, E. Kushilevitz, M. Naor, and N. Nisan. Amortized communication complexity. *SIAM Journal on Computing*, 24(4):736–750, 1995.
- [22] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In *Proc. 28th International Colloquium on Automata, Languages and Programming*, pages 927–938. Springer-Verlag, 2001.
- [23] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002. Earlier version in FOCS '99.
- [24] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology — EUROCRYPT '04*, LNCS 3027, pages 1–19. Springer-Verlag, 2004.
- [25] D. Gavinsky, J. Kempe, and R. de Wolf. Quantum communication cannot simulate a public coin. <http://xxx.lanl.gov/abs/quant-ph/0411051>, 2004.
- [26] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Proc. 32nd International Colloquium on Automata, Languages and Programming*, pages 803–815, 2005.
- [27] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. Computer and System Sciences*, 60(3):592–692, 2000. Earlier version in STOC '98.
- [28] O. Goldreich. *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge University Press, 2004.

- [29] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. 19th Annual ACM Symposium on the Theory of Computing*, pages 218–229. ACM Press, 1987.
- [30] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Computer and System Sciences*, 28:270–299, 1984.
- [31] S. Halevi, E. Kushilevitz, R. Krauthgamer, and K. Nissim. Private approximations of NP-hard functions. In *Proc. 33th Annual ACM Symposium on the Theory of Computing*, pages 550–559, 2001.
- [32] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 189–197, 2000.
- [33] P. Indyk and D. P. Woodruff. Polylogarithmic private approximations and efficient matching. In *Proc. Third Theory of Cryptography Conference*, pages 245–264, 2006.
- [34] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *Proc. 36th Annual ACM Symposium on the Theory of Computing*, pages 262 – 272, 2004.
- [35] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.
- [36] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004. Earlier version in STOC '01.
- [37] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.
- [38] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. In *Proc. 27th Annual ACM Symposium on the Theory of Computing*, pages 398–406, 1995.
- [39] J. Katz, R. Ostrovsky, and A. Smith. Round efficiency of multi-party computation with a dishonest majority. In *Advances in Cryptology — EUROCRYPT '03*, LNCS 2656, pages 578–595. Springer-Verlag, 2003.
- [40] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.
- [41] E. Kushilevitz and R. Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997.
- [42] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. Earlier version in STOC '98.
- [43] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology*, 16(3):143–184, 2003. Earlier version in Crypto '01.
- [44] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002. Earlier version in Crypto '00.
- [45] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In J. Zhou and J. Lopez, editors, *the 8th Information Security Conference (ISC'05)*, volume 3650 of LNCS, pages 314–328. Springer-Verlag, 2005.
- [46] E. Mann. Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology, Haifa,, 1998.
- [47] S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology — CRYPTO '91*, LNCS 576, pages 392–404. Springer-Verlag, 1991.
- [48] H. Minc. Permanents. In *Encyclopedia of Mathematics and its Applications*, volume 6. Addison-Wesley, 1982.

- [49] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. on Computing*, 22(4):838–856, 1993. Earlier version in STOC '90.
- [50] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. 33th Annual ACM Symposium on the Theory of Computing*, pages 590–599, 2001.
- [51] M. Naor and B. Pinkas. Computationally secure oblivious transfer. *J. Cryptology*, 18(1):1–35, 2005. Earlier version in “Oblivious transfer and polynomial evaluation”, STOC '99.
- [52] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th Annual ACM Symposium on the Theory of Computing*, pages 232–241, 2004.
- [53] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.
- [54] J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology — ASIACRYPT '98*, LNCS 1514, pages 357–371. Springer-Verlag, 1998.
- [55] A. Yao. Protocols for secure computation. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.
- [56] A. Yao. On the power of quantum fingerprinting. In *Proc. 35th Annual ACM Symposium on the Theory of Computing*, pages 77–81, 2003.

## A General Definition of Secure Computation

In this section, we sketch the standard simulation-based approach for defining secure computation. This definition generalizes Definition 3 in that it addresses also the case of multiple parties and the case of an active (malicious) adversary. Our definition refers to the stand-alone setting (i.e., it does not consider protocol composition) and refers to the case of a non-adaptive adversary, who picks the set of corrupted parties in advance. We do not address here the relaxed notion of “security with abort,” which is necessary when the adversary is active and may corrupt at least half of the parties. We refer the reader to [12, 28, 13] for more general and detailed definitions.

Let  $\pi$  be an  $m$ -party protocol and let  $\mathcal{A}$  be an adversary corrupting at most  $t$  parties. The following definition compares the interaction of the adversary in the real-life protocol with the interaction of an adversary with an *ideal process* for evaluating the target function  $f$ .

**Real-life model.** The interaction of the adversary in the real-life model is captured by a random variable  $\text{REAL}_{\pi, \mathcal{A}}(\mathbf{x})$ , set to the view of  $\mathcal{A}$  when attacking the execution of  $\pi$  on input  $\mathbf{x}$ , concatenated with the outputs of the uncorrupted parties and their identities. The adversary’s view includes all inputs, random inputs, and messages viewed by corrupted parties. The concatenation of this view with the outputs on non-corrupted parties serves two purposes. First, it captures the information that the adversary may learn about the outputs of uncorrupted parties. Second, it captures the correctness requirement of the protocol (possibly in the presence of an active adversary who tries to alter the outputs of uncorrupted parties).

**Ideal process.** The *ideal process* is parameterized by a target function  $f$ , which may be a general, possibly randomized, mapping from  $m$  inputs to  $m$  outputs. In our context, it is convenient to restrict  $f$  to be a deterministic, single-output function. An adversary  $\mathcal{A}'$  corrupting the ideal process is referred to as an *ideal-process adversary* or a *simulator*. The ideal process proceeds as follows. First,  $\mathcal{S}$  decides on a set  $T$  of at most  $t$  parties to corrupt, where  $t$  is the given security threshold. If the adversary is active, it may first modify the inputs of the parties it corrupts based on their observed values. Subsequently, all parties send their inputs to a trusted party, who evaluates the function  $f$  and hands each of its outputs to the corresponding party. (If  $f$  is a single-output randomized function, then our convention is that each party receives an identical instance of its output.) Based on the inputs and outputs of corrupted parties, the adversary produces some output, which is supposed to emulate the transcript of the real-life protocol. The interaction of the adversary  $\mathcal{A}'$  with the  $f$ -ideal process on input  $\mathbf{x}$  is captured by a random variable

$\text{IDEAL}_{\pi, \mathcal{A}', f}(\mathbf{x})$ , containing the adversary’s output concatenated with the outputs of uncorrupted parties and their identities.

We now formalize our definition of a secure protocol.

**Definition 10** *A protocol  $\pi$  is said to be a perfectly/statistically/computationally  $t$ -secure protocol for  $f$ , if for any adversary  $\mathcal{A}$  corrupting at most  $t$  parties in the real-life model, there exists a probabilistic polynomial-time simulator  $\mathcal{A}'$  corrupting at most  $t$  parties in the ideal process, such that*

$$\{\text{REAL}_{\pi, \mathcal{A}, f}(\mathbf{x})\}_{\mathbf{x} \in X} \equiv \{\text{IDEAL}_{\pi, \mathcal{A}', f}(\mathbf{x})\}_{\mathbf{x} \in X},$$

where “ $\equiv$ ” denotes perfect/statistical/computational indistinguishability. In the case of computational security, the adversary  $\mathcal{A}$  is restricted to probabilistic polynomial time. In the other two cases, it may be computationally unbounded and the time resources of  $\mathcal{A}'$  are allowed to be polynomial in those of  $\mathcal{A}$ .

In this paper, we consider by default the case of a computationally 1-secure 2-party protocol, whose security holds against a passive adversary.

## B A Liberal Definition of Secure Approximation

In this section, we detail the general formulation of the liberal definition of secure approximations discussed in Section 3.1.

The liberal definition of secure approximations modifies the simulation-based definition of secure (exact) computation from Appendix A as follows. The real model remains unchanged. The  $f$ -ideal model is modified to what we call the  $(f', \hat{f})$ -ideal model. For a single-output, possibly randomized function  $\hat{f}$ , the corresponding random variable  $\text{IDEAL}_{\pi, \mathcal{A}', f', \hat{f}}(\mathbf{x})$  is defined similarly to  $\text{IDEAL}_{\pi, \mathcal{A}', f}(\mathbf{x})$  from Appendix A, with the following modification. Instead of sending the values of the single function  $f$  to all parties, the trusted party evaluates both  $f'$  and  $\hat{f}$  on the inputs it receives and sends the two values to all parties. All uncorrupted or passively corrupted parties output the value  $\hat{f}$  alone.

The function  $f'$  models the information that we allow the adversary to learn, whereas  $\hat{f}$  captures the correctness requirement for the outputs of uncorrupted parties. Our liberal definition lets  $f'$  be the same as the target function  $f$  by default. This is philosophically justified by the fact that when approximating a function  $f$ , one is implicitly willing to pay the privacy compromise implied by the knowledge of  $f$ . However, in some cases it may be desirable to choose  $f'$  so that it reveals strictly less information than  $f$ ; the above formulation provides a convenient means for formalizing the type of “extra security” provided in such cases. Taking  $f' = \hat{f}$  yields precisely the strict notion of Definition 6.

We now give the general formulation of the liberal definition of a secure approximation protocol.

**Definition 11 (secure approximation: general liberal definition)** *A protocol  $\pi$  is said to be a perfectly/statistically/computationally  $t$ -secure  $\mathcal{P}$ -approximation protocol for  $f$  in the liberal sense, if there exists a functionally private  $\mathcal{P}$ -approximation  $\hat{f}$  of  $f$  such that the following holds. For any probabilistic polynomial-time adversary  $\mathcal{A}$  corrupting at most  $t$  parties in the real-life model, there exists a probabilistic polynomial-time simulator  $\mathcal{A}'$  (corrupting at most  $t$  parties) in the ideal process, such that*

$$\{\text{REAL}_{\pi, \mathcal{A}, f}(\mathbf{x})\}_{\mathbf{x} \in X} \equiv \{\text{IDEAL}_{\pi, \mathcal{A}', f, \hat{f}}(\mathbf{x})\}_{\mathbf{x} \in X},$$

where “ $\equiv$ ” denotes perfect/statistical/computational indistinguishability.

In this paper, we consider by default the case of a computationally 1-secure 2-party protocol whose security holds against a passive adversary.