Cryptography from Anonymity

Yuval Ishai* Eyal Kushilevitz† Rafail Ostrovsky‡ Amit Sahai§

March 2, 2006

Abstract

There is a vast body of work on *implementing* anonymous communication. In this paper we study the possibility of using anonymous communication as a *building block*, and show that one can leverage on anonymity in a surprising variety of cryptographic contexts. Our results go in two directions.

- **Feasibility.** We show that anonymous communication over *insecure* channels can be used to implement unconditionally secure point-to-point channels, broadcast, and general multi-party protocols that remain unconditionally secure as long as less than *half* of the players are *actively* corrupted.
- **Efficiency.** We show that anonymous channels can yield major efficiency improvements for several natural secure computation tasks. In particular, we present the first solution to the problem of private information retrieval (PIR) which can handle multiple users while being close to optimal (in an amortized sense) with respect to *both* communication and computation.

*Computer Science Department, Technion.E-mail: yuvali@cs.technion.ac.il. Research supported by grant 36/03 from the Israel Science Foundation and grant 2004361 from the U.S.-Israel Binational Science Foundation.

†Computer Science Department, Technion. E-mail: eyalk@cs.technion.ac.il. Research supported by grant 36/03 from the Israel Science Foundation.

‡Department of Computer Science, UCLA. E-mail: rafail@cs.ucla.edu.

§Department of Computer Science, UCLA. E-mail: sahai@cs.ucla.edu. Research supported by grant 2004361 from the U.S.-Israel Binational Science Foundation.

1 Introduction

There are many scenarios in which anonymous communication can be implemented at low cost, either via physical means (e.g. in wireless networks, or small wired networks) or by means of special-purpose protocols. Indeed, a lot of systems work has been done on implementing anonymous communication (cf. [1, 12, 50, 7] and references therein). Anonymizing web browsers and anonymous email accounts are already widely available. In this work, we ask the question: If anonymity is already out there, can we harness its power for other purposes? To what extent can anonymity be used as a *building block* for obtaining better solutions to other important cryptographic tasks? We elaborate on this question below.

Anonymity vs. privacy. Anonymous communication allows users to send messages to each other without revealing their identity. However, in contrast to popular misconception, anonymity is far from answering all concerns of "privacy".[1] Conceptually, anonymity is aimed at hiding *who* performs some action whereas full privacy requires to additionally hide *what* actions are being performed. In the context of distributed computation, anonymity allows to hide which users holds which local inputs, whereas privacy requires to hide all information about the inputs except what follows from the outputs. In a sense, the relation between anonymity and privacy is analogous to the relation between unpredictability and indistinguishability: while the former notions of security might be sufficient for some applications, they are generally considered inadequate; in particular, they are vulnerable to attacks that exploit a-priori information about the secrets.

The aim of the current work is to study the extent to which anonymity can be useful as a *primitive*. Can the gap between hiding the *Who* and the *What* be closed at a small additional cost? Can anonymity be exploited in other cryptographic contexts, beyond those that involve privacy?

A toy example. As a simple motivating example, consider the following scenario. Two players, A and B, wish to agree on an unconditionally secret random bit (a "key"). Their only means of communicating is by posting anonymous messages on a *public* internet bulletin board. (In this example we assume that the board operator as well as all of the users are "honest but curious".) The key agreement protocol proceeds as follows. Each player $P \in \{A, B\}$ independently picks a random 50-bit integer r_P, and posts the message ("AB", r_P) on the board. The common bit is taken to be 0 if $r_A > r_B$ and 1 if $r_A < r_B$. (In the unlikely event of a tie, the protocol aborts.) Note that since each player P knows its integer r_P they can both compute the (same) common bit, whereas other users (as well as the board operator) cannot distinguish r_A from r_B and thus learn nothing about the common bit. Of course, the 1-bit key can now be used by A and B to communicate a bit with unconditional secrecy using the public bulletin board.

1.1 Our Contribution

We demonstrate the usefulness of our "cryptography from anonymity" paradigm in two settings: (1) Establishing *feasibility* results for traditional cryptographic tasks unconditionally, based solely on the assumption of anonymous channels. (2) Showing that anonymous channels can lead to much more *efficient* solutions to several cryptographic problems. We now provide a detailed account of both types of results.

1.1.1 Feasibility results based on anonymity

We start by studying which tasks can be implemented with *unconditional* security based on anonymous communication. To this end, we consider the following weak model of anonymity over *public* channels.

[1] The term "privacy" has different interpretations. Our usage of this term below follows the common terminology in the literature on cryptographic protocols.

In each round each player can send a message to a chosen destination. The adversary can learn, for every player in the network (including uncorrupted players), the *multiset* of all messages received by that player. The adversary does *not* learn the identity of the sender, except when the sender itself is corrupted. In addition to such anonymous channels, we also assume that the players can communicate via *authenticated* but public point-to-point channels.

One of the challenges that need to be faced when attempting to exploit such a network is the fact that anonymity can also serve as a *shelter* for malicious players. For instance, if the protocol instructs only some strict subset of the players to (anonymously) send messages to the same receiver, malicious players outside this set can interfere by sending their own messages. Note that one cannot make a direct use of authentication to distinguish "legitimate" messages from illegitimate ones, as this would violate anonymity.

In the above model, we show how to realize the following primitives:

1. Secure point-to-point communication with unconditional security against an arbitrary number of malicious players. (This protocol strengthens the simple key agreement protocol described above in that its security is not restricted to the case of "honest but curious" players.)

2. General multi-party protocols with unconditional security against any minority of malicious players. (Note that such protocols cannot be based on secure point-to-point channels alone even for simple functionalities such as broadcast.)

As an intermediate step towards establishing the above results, we construct a stronger form of "private" anonymous channels out of the basic public anonymous channels defined above. Such private anonymous channels allow the adversary to learn only messages sent or received by corrupted players. The equivalence between private and public anonymity is of independent interest, since there is no obvious construction of private anonymity from public anonymity even if one is additionally given secure (but non-anonymous) point-to-point channels.

The above results do not rule out the possibility that anonymous communication can be used to solve *any* cryptographic task with an arbitrary level of security. However, we show that this is not the case: anonymity cannot be used to build an oblivious transfer protocol with unconditional security against half or more of the players. Thus, our general feasibility result achieves an optimal level of security.

1.1.2 Efficiency improvements based on anonymity

We now turn the attention to the question of *efficiency*, attempting to identify natural cryptographic tasks for which anonymity can give rise to substantial efficiency gains. In contrast to the feasibility results discussed above, here we are not restricting ourselves to unconditional results. In particular, we would like to improve over the best known solutions under *any* cryptographic assumption. A key observation that underlies our protocols in this setting is that *local randomization* of inputs, via secret-sharing, when combined with the *global mixing* of the shares, provided by anonymity, allows to keep the inputs private and at the same time still allow to carry out some useful computations on the inputs. We elaborate below.

"Split and Mix" approach. Consider a scenario in which several clients want to access or query a central server without revealing their sensitive data to the server. For instance, the clients may want the server to compute some global function of their joint queries (e.g., average salary of employees), or alternatively to respond to each query separately (as in the case of retrieving data from a central database). As discussed above, anonymity alone does not provide a good solution to this problem. While the mixing effect achieved by anonymity eliminates some information about the query made by each client, most information remains. Our key idea is to boost the effect of anonymity by using local randomization as a *catalyst*. Specifically,

we first let each client locally *split* its query into few randomized sub-queries (e.g., via the use of secret-sharing), and then let the clients *mix* all their sub-queries by anonymously sending them to the server. (Here we assume that all sub-queries are mixed together, so that the server cannot tell whether two sub-queries were sent by the same client.) The hope is that the mixed sub-queries can *totally* eliminate unnecessary information about the queries (either statistically or computationally). Moreover, the splitting should be done in a way that allows to carry out the desired functionality on the original queries based on the mixed sub-queries. We stress that neither mixing nor splitting alone can provide an adequate level of privacy; but as it will turn out, their combination can be surprisingly powerful. We demonstrate the usefulness of this "split and mix" approach in several contexts, described below.

Non-interactive private statistics. We consider the case where each of two or more clients holds an m-bit input, and they wish to reveal to a server the *sum* of their inputs (and nothing else) by simultaneously sending anonymous messages to the server. (Note that without anonymity, it is impossible to solve this problem in a completely non-interactive way as we require, regardless of efficiency.) A simple but inefficient solution is to let each client i, holding an integer x_i, anonymously send x_i distinct dummy messages to the server. The server can now compute the sum of all inputs by simply counting the number of messages it received. This simple solution provides perfect privacy, but does not scale with the bit length m. Towards a more efficient solution, we use the split and mix approach in the following way. Each client locally splits its input into $O(m)$ shares via the use of *additive* secret-sharing, and anonymously sends its shares to the server. The server can now recover the sum of the inputs by adding up all the shares it received. We show that with this choice of parameters, the mixed messages reveal only a negligible amount of information about the inputs (other than their sum). This basic integer summation protocol can be used as a building block for computing other kinds of useful statistics on distributed data. We also show an application of this protocol in a general context of two-party computation, where each client wants to privately compute a function of its own input and the server's input.

Optimal amortized PIR. In the problem of Private Information Retrieval (PIR) [15, 38] the server holds a (large) database of size n, and each client wants to retrieve a specific item from this database while hiding what it is after. This can be trivially done by having the server communicate the entire database to each client, but this solution is prohibitively expensive. In recent years, there has been a significant body of work on improving the communication complexity of PIR, either in the above single-server scenario [38, 10, 39, 25] or using multiple servers [15, 3]. Given the low (and essentially optimal) communication complexity of the best known PIR protocols, the efficiency bottleneck shifts to the *local computation* performed by the server. Indeed, it is not hard to see that even in the case of a single query, the server must read every bit of the database in order for privacy to be maintained. Thus, the best one could hope for is to amortize this cost over multiple queries.

The question of amortizing the computational cost of PIR has been previously considered in [4, 33]. However, all previous solutions to this problem either require multiple servers (and totally fail to protect against colluding servers) or only allow to amortize the cost of different queries that originate from the same client.[2] A major open problem is this area is to obtain solutions to PIR that are (close to) optimal with respect to *both* communication and computation even in the case where queries originate from different clients. In this work we suggest a solution to this problem using (two-way) anonymous communication, and assuming that the cost is amortized over many clients. Our solution applies the "split and mix" technique as follows. First, each client randomizes its query into a small number of sub-queries by simulating an appropriate

[2] In [4] it was demonstrated that the computational cost of PIR can be *slightly* amortized even in the case of queries that originate from different clients.

multi-server PIR protocol (with privacy threshold equal to the security parameter). Then, all sub-queries are anonymously sent to the server, who responds to each sub-query separately without knowing which client it originates from. Each client can recover the answer to its query from the server's answers to its sub-queries as in the underlying multi-server PIR protocol. The computation in this protocol can be amortized by choosing the parameters so that the space of all possible sub-queries is of polynomial size. This enables to precompute the answers to all possible sub-queries.

The security of our PIR protocol relies on an intractability assumption related to the hardness of interpolating *noisy* low-degree curves in a low-dimensional space. A similar assumption for the two-dimensional case has been previously used by Naor and Pinkas [42, 43]. Roughly speaking, the original assumption from [42, 43] asserts that noisy two-dimensional curves cannot be interpolated in a "much better" way than using the Guruswami-Sudan list decoding algorithm [28]. Our generalized assumption asserts that in the low-dimensional case, one cannot do "much better" than the Coppersmith-Sudan algorithm [17] (which extends [28] to the multi-dimensional case). We note that this assumption does not seem to be affected by recent progress in the field of list-decoding [46, 27]. It is also instructive to note that this assumption (as well as the one from [42, 43]) is not known to imply public-key encryption, let alone PIR, in the standard setting. Accordingly, for the protocol to be secure, the *total* communication with all clients must be larger than the database size. Still, when the number of clients is large, the *amortized* communication (and computation) per client becomes low.

Finally, we observe that the special structure of the above PIR protocol allows to distribute the role of the server between many different users without compromising efficiency or privacy. This gives rise to conceptually attractive distributed storage systems (e.g., in a peer-to-peer environment) which are simultaneously close to optimal with respect to communication, computation, load balancing, storage, robustness, and privacy.

On relaxing the anonymity assumption. While we assume for simplicity that the network provides perfect anonymity, this assumption can be relaxed. In fact, our protocols only require "sufficient uncertainty" about the origins of messages to maintain their full cryptographic security, at a modest additional cost. For instance, consider the following generalization of the simple key agreement protocol described above. Instead of posting a single message on the bulletin board, each player posts m random and independent messages. (To prevent the adversary from linking different messages sent by the same player, the timing of the messages is randomly spread within some fixed time interval.) As a result, both players learn a random subset $S \subseteq \{1, 2, \ldots, 2m\}$ of size m, consisting of the positions of messages posted by A. Now, suppose that an adversary can partially break the anonymity and obtain some partial information about the set S. As long as there is sufficient uncertainty about S from the adversary's point of view, a random secret key can be obtained by making a standard use of privacy amplification [6]. Specifically, suppose that the adversary cannot *precisely* trace *all* messages to their origin, except with 2^{-k} success probability. (Still, it might be the case that the adversary can trace 99% of the messages with absolute certainty.) Applying a random (pairwise independent) hash function to the set S, the players can agree on a key of length $k' = k - \sigma$ which given the adversary's view is within statistical distance $2^{-\Omega(\sigma)}$ from random. A similar approach can be applied to obtain robust versions of other protocols we present. (In fact, such robustness is already built into some of the protocols, like the private sum protocol described above.) Thus, our approach is quite insensitive to imperfections of the underlying network, and can be applied even in more realistic scenarios where only some crude form of anonymity is available.

1.2 Related Work

A variant of the toy example presented above (for key agreement using anonymity) was suggested by Alpern and Schneider [2]. In contrast, we achieve key agreement in the presence of *malicious* parties, a problem that was posed and left open in [2]. The related problem of obtaining key agreement from *recipient anonymity*, which hides the identity of the receiver rather than that of the sender, was considered in [47]. Anonymous communication has also been exploited in the context of practically oriented applications such as voting [13] and electronic cash [52].

Our work can be cast in the setting of investigating secure reductions between different multiparty functionalities: An anonymous channel can be modelled by such a functionality, and we are investigating what other functionalities can be realized using this functionality (and how efficiently). This area has a rich history (see [36, 37, 22, 29, 5, 24, 41] and references therein). However, most of the work in this area has been restricted to the two-party setting, and known results for the multi-party setting are not general enough to apply to the case of the anonymity functionality. Moreover, relatively little attention has been paid to the *efficiency* of such reductions.

Pfitzmann and Waidner [48] use a variant of private anonymous communication as an intermediate step for obtaining highly resilient broadcast protocols (in a model that allows broadcast in a precomputation phase). We rely on their protocol as a step in obtaining our feasibility result for MPC with honest majority. We stress that the protocol of [48] requires the use of *private* anonymity, and thus does not directly imply an implementation of broadcast in our basic model of non-private anonymity. For self-containment, we also present a simpler alternative to the use of the protocol of [48] in our context.

Finally, our approach is also reminiscent of recent work on data privacy through data *perturbation* (cf. [19, 14, 53]). These works examine privacy through "blending in with the crowd" [14], obtained via the perturbation of data revealed to the adversary. In our work, we also examine how privacy can be achieved by blending in with the crowd, but via *mixing* rather than perturbation.

2 Preliminaries

Notation. We denote by $[n]$ the set $\{1, 2, \ldots, n\}$ and by $\binom{[n]}{k}$ the collection of all subsets of n of size k. We use $\log$ to denote $\log_2$, the logarithm to the base 2. We denote by $SD(X, Y)$ the statistical distance between probability distributions X, Y, defined as the maximal advantage of a (computationally unbounded) distinguisher A in telling X and Y apart. That is, $SD(X, Y) = \max_A |\Pr[A(X) = 1] - Pr[A(Y) = 1]|$. We denote by $H_\infty(X)$ the min-entropy of X defined by $H_\infty(X) = \min_x(-\log \Pr[X = x])$, where the minimum is taken over all x in the support of X.

We rely on the following version of the Leftover Hash Lemma [31], asserting that a pairwise independent hash function can be used as a strong *randomness extractor* [45].

Lemma 2.1 (Leftover Hash Lemma) *Let $\mathcal{H}$ be a family of pairwise independent hash functions $h : A \to B$ and let H denote a uniformly random $h \in_R \mathcal{H}$. Let V be a random variable over A such that $H_\infty(V) \geq \log |B| + \sigma$ and U be a random variable uniformly distributed over B, independently of H. Then,*

$$SD((H, H(V)), (H, U)) \leq 2^{-\Omega(\sigma)}.$$

2.1 Network Model

In what follows, we define the network model that we use in this paper. Then, we discuss the various notions of anonymity that we consider. We denote by N the number of players in the network. In all variants of anonymity, we start with a standard network that allows, for each pair of players P_i, P_j, communication

over an *insecure* point-to-point authenticated channel. (By "insecure" we mean that all messages sent over the channels are viewed by the adversary.) Then, we augment this network by some form of anonymous point-to-point communication.

The main type of anonymity we consider in this work is that of *sender anonymity*, where the sender of each message is hidden from the adversary (but both its content and its designated receiver are known). We distinguish between the basic *one-way anonymous channel*, where the receiver of a message has no way to "answer" a message, and *two-way anonymous channel* where the anonymity mechanism allows "feedback", i.e., answering the sender of a message while still keeping the sender's identity secret.

More specifically, our default model allows only *one-way* anonymous communication: at each round, each player P_i may send a single message to some player P_j.[3] The adversary learns the contents of *all* messages exchanged between the players, including the destination of each message but not its source. We denote this basic anonymity functionality by Anon. Alternatively, one could assume that the adversary only learns messages received by corrupted players; we call such a primitive *private-anonymous channel* and denote the corresponding functionality by PrivAnon. We will show in Section 3.1 that the two variants are equivalent. We stress that in both cases, the adversary cannot learn any information about the sources of messages that originate from uncorrupted players. A formal definition of the functionalities Anon and PrivAnon appear in Appendix A. Our definitions allow the adversary to be *rushing*, namely to choose the messages it sends depending on messages received by corrupted players.

We also consider two-way anonymous communication. In this model, each invocation consists of two rounds, allowing each player P_i to anonymously send a message to any player P_j and to receive a reply to its message. For each such message-reply pair sent from player P_i to P_j and back, the adversary learns the identity of the destination player j, but not the identity of i; it can also tell that the second message is a reply to the first. Most physical or algorithmic implementations of anonymity support this two-way communication (cf. [12, 7, 51]). As before, this can be formalized as an interactive functionality, as described in Appendix A.

Note that it is easy to implement two-way anonymous channels from one-way anonymous channels: each sender sends its message m to the desired receiver (using one-way anonymous channel); each receiver replies (using public authenticated channels) *to all players* with a list (m_i, m'_i), where m'_i is the answer to message m_i. (Note that the answer m'_i should depend only on the content of the query m_i and not on the identity of the sender; thus, the receiver only needs to provide one answer to duplicate queries.) The problem with this reduction is that it is generally inefficient. Thus, in our applications that rely on two-way anonymous communication we assume that the network provides an efficient direct realization of this primitive.

Finally, in some of our application we will consider scenarios where the players are partitioned into two or more *clients* and a single *server*, so that each client only needs to interact with the server and not with other clients. Clearly, the above definitions apply to such a setting as well.

Secure reductions. Our results can be viewed as reductions between cryptographic primitives; namely, they show how to implement a certain primitive (or functionality) g given a black-box access to a functionality f, where f is typically an anonymity functionality. By a t-secure reduction from g to f we refer by default to a *statistically* t-secure protocol for g in the so-called f-hybrid model (i.e., in a model where players have access to a trusted oracle computing f). For simplicity we consider *non-adaptive* adversaries, who choose the set of corrupted players in advance. For a formal definition of "statistically t-secure protocols", see [11, 26].

[3] We can extend this basic definition by allowing each player to send up to λ messages at each round, for some parameter λ. Note that without such a bound the adversary may "flood" the network with anonymous messages.

3 Feasibility Results Based on Anonymity

In this section we present unconditionally secure implementations of several cryptographic primitives based on anonymous communication over public channels.

3.1 Private Anonymity from Public Anonymity

We start by showing how to reduce the private anonymity functionality to the basic anonymity functionality defined in Section 2.1. This will serve as a stepping stone towards implementing other primitives. Recall that the private anonymity functionality PrivAnon differs from the basic one in that the adversary can only learn the set of messages received by each corrupted player, and learns nothing about messages exchanged between uncorrupted players. Thus, PrivAnon is defined similarly to our basic anonymity functionality Anon, except that the adversary only learns the contents of the messages sent to corrupted players.

The goal of securely reducing PrivAnon to Anon (i.e., constructing private anonymous channels from non-private anonymous channels) is nontrivial even if we were additionally given secure (non-anonymous) point-to-point channels. Indeed, there is no obvious way to combine the advantages of insecure but anonymous channels and secure but non-anonymous channels. Instead, we suggest the following direct implementation of PrivAnon based on Anon.

Assume for now, that there are only 3 players, A, B (senders) and R (receiver); See Remark 3.2 below for the generalization to the N-party case. Recall that our basic model assumes non-private anonymous channels (as defined by Anon) and authenticated non-private channels. We wish to construct a protocol allowing A or B to send a message to R with the following properties:

1. If A and B are honest then their anonymity is preserved.
2. If A and R are honest then privacy of the message is preserved when A sends an anonymous message to R.

Below, we write $AE_K(m)$ to denote a (statistically secure) one-time authenticated encryption of the message m using the key K. Such an encryption can be decrypted and authenticated using the secret key K. It can be implemented with unconditional security by using a one-time pad for encrypting and an unconditionally-secure MAC for authenticating. The protocol is as follows:

1. Repeat the following $3k$ times sequentially (each is referred to as a "session"):
   (a) Each of A, B and R sends a k-bit number to R, using the anonymous (non-private) channels.
   (b) R considers the numbers received in the previous step, ignoring repetitions, and chooses 2 out of these numbers, including its own number. R sends these 2 numbers in lexicographic order to both players A and B via authenticated channels. The order of these 2 numbers defines a (secret) bit between R and either A or B according to who's number is chosen. (With overwhelming probability, all honest players choose distinct numbers; the adversary, being rushing, may duplicate numbers selected by honest players.)
2. Each of A and B sends to R, via non-private anonymous channels, a list of the first k session numbers in which they obtained shared bits. (Note that, with overwhelming probability, each of A and B obtained at least k shared bits.) This results in the definition of two k-bit secret keys K_A and K_B. The receiver, R, knows both keys, but does not know which of them belongs to A and which belongs to B. (i.e., from R's perspective they are two keys $\{K_1, K_2\} = \{K_A, K_B\}$).

3. To send private anonymous messages m_A and m_B to R, the two players A and B send $AE_{K_A}(m_A)$ and $AE_{K_B}(m_B)$, respectively, via the non-private anonymous channel to R. The receiver R authenticates and decrypts each message using the keys K_1 and K_2 (this allows identifying key corresponding to the message)[4].

Theorem 3.1 *The above protocol defines a statistically-secure reduction from the private anonymity functionality,* PrivAnon*, to the basic non-private anonymity functionality,* Anon.

Proof sketch: If any two players are dishonest, the protocol does not need to provide any security guarantees to the remaining honest player.

If the receiver R is dishonest, but both A and B are honest, then we must guarantee anonymity of A and B. This follows from the symmetry of the protocol.

If the receiver R and one sender, A, are honest, but the other sender B is dishonest, we must guarantee privacy and integrity of A's message to R. By the properties of the authenticated encryption AE, this is guaranteed as long as we can prove that B's view contains no information about the key K_A established between A and R. This is demonstrated by giving a family of $3k$ bijections on the randomness used by honest players A and R, that preserve B's view. Bijection i works by swapping the random numbers generated by A and R in the i'th invocation of Step 1 (but leaving all other random choices intact). Since B only observes the *set* of values sent anonymously to R (which includes of messages from A and R itself), the view before the application of any bijection is identical to the view after the application of the bijection. The existence of these bijections shows that all possible values of K_A are equally likely for any view of the adversary. □

Remark 3.2 The following changes should be applied to the above protocol, when dealing with N-players network (a receiver R and $N-1$ potential senders). In such a case, we increase the number of "sessions" to $2Nk$ which guarantees that the numbers chosen by each sender appear in the pairs selected by R at least k times (also, k is sufficiently large so that in each session all the numbers chosen by honest players are distinct). This allows each sender to send a list of k session numbers for which it knows the corresponding secret bit.

3.2 Secure Point-to-Point Channels

In this section, we show how to use anonymous channels to construct secure point-to-point channels. In fact, it suffices to show how to construct a key-agreement protocol over anonymous channels and, in light of the previous sub-section, it suffices to do so using *private* anonymous channels.

The simple key agreement protocol presented in the Introduction (as well as a similar protocol from [2]) allows A and B to agree on a secret random key r by posting messages on an anonymous bulletin board. This protocol assumes that the bulletin board operator is semi-honest, in the sense that it accurately posts the received messages on the board. Furthermore, the protocol implicitly assumes that no other users are present in the system to interfere with the messages sent by A and B.

We now describe a key-agreement protocol that works in our standard model, namely where the players A, B are just two players in a network of N players. The main difficulty in utilizing the anonymity in this case is that when one of the players needs to send an anonymous message, then corrupted players may attack the protocol by also sending messages over the anonymous channel. Our protocol prevents this attack, even if all other $N-2$ players are malicious, via a simple reduction to private anonymity (i.e., we assume the availability of private anonymous channels, as constructed above).

[4]Note that these keys should not be used more than once; otherwise R can correlate coming messages.

1. A sends a random k-bit key, K_A, to B via a private-anonymous channel.
   [Note that malicious players may also send to B anonymous k-bit messages.]

2. B sends to A the list of all keys received in step (1), via a private anonymous channel.
   [Note that malicious players may also send to A such lists.]

3. From all lists received in step (2), player A finds the single list L which includes her original key K_A (if there is no such list then A aborts), and sends over a public-authenticated channel the position of K_A in this list.

Theorem 3.3 *The above protocol defines a statistically N-secure reduction from the key agreement functionality to the private anonymity functionality* PrivAnon.

Proof sketch: We sketch the main properties of the protocol. Note that the only interesting case is where both A and B are honest and the dishonest players may try to either prevent A and B from reaching an agreement or to obtain information about the key that A and B agree on. Observe that, except with exponentially small probability, exactly one of the lists received by A in step (2) contains its own key K_A; this list, L, if the list originates from B (we rely on the anonymous channel being *private* which prevents, for example, replay attacks). Therefore, the protocol terminates with high probability and, moreover, since both K_A and L are sent over a *private* anonymous channel, then the value of K_A remains hidden from all other players. □

In Appendix B, we discuss an alternative means for obtaining key-agreement protocols in our setting via a reduction to the problem of key agreement using a "deck of cards" [21] (see also [40]).

3.3 General Secure Multiparty Computation

We turn to the question of basing general secure multiparty computation (MPC) on anonymity. To this end, we first implement secure *broadcast* based on anonymity. Combined with the implementation of secure channels from the previous section, one can then apply known MPC protocols (e.g., [49] or [18]) and obtain general t-secure MPC with $t < N/2$. Given Theorem 3.1 it therefore suffices to reduce the *broadcast* functionality to *private* anonymity. It turns out that this type of reduction is implicit in the work of Pfitzmann and Waidner [48] on obtaining fully resilient broadcast using preprocessing.

Lemma 3.4 (implicit in [48]) *There is a statistically N-secure reduction of the broadcast functionality to* PrivAnon.

The reduction from [48] is rather complex and is broken into several stages.[5] For self-containment, we present in Appendix C a simpler reduction of this type. This alternative reduction is only t-secure for $t < N/2$, but this is all we need for obtaining the following main result of this section:

Theorem 3.5 *For any N-party functionality f, there is a statistically t-secure reduction of f to* Anon *for any $t < N/2$.*

[5] Specifically, first a variant of anonymity that is similar to (but is slightly weaker than) our PrivAnon primitive is used to construct *pseudo-signatures*, an information-theoretic analog of digital signatures. Then, a broadcast protocol is obtained based on the pseudo-signatures. We note that the anonymity primitive itself is realized using secure channels and broadcast (which are assumed in [48] to be available at a preprocessing stage).

3.4 Limitations of Anonymity

We have shown that any N-party functionality can be reduced to anonymity with statistical t-security provided that $t < N/2$. We now argue that this bound on t is tight.

We start by showing that in the semi-honest model, anonymity is *equivalent* to secure channels.

Claim 3.6 *In the semi-honest model, anonymous channels are equivalent to secure point-to-point channels with respect to statistically n-secure reductions.*

Proof sketch: A reduction from secure channels to anonymity was shown above. In the other direction, consider the following protocol for implementing (private) anonymous channels using secure channels. Suppose that all possible messages are from the domain $[\ell]$, for some integer ℓ.[6] For each possible destination P_j, each player P_i that wishes to send a message $m_i \in [\ell]$ to P_j uses as input to the protocol the ℓ-bit unit vector e_{m_i}; each player P_i that has no message to send to P_j uses, as its input to the protocol, the length-ℓ all-zero vector. Next, the players apply an N-private protocol that computes the coordinate-wise sum of the n inputs (modulo N). Finally, note that this sum reveals to P_j the multiset of messages sent to him by the players, but it reveals no information regarding which source contributed which value. □

Since Oblivious Transfer (OT) cannot be implemented with statistical t-security in the secure channels model when $t \geq N/2$ [16], it follows that OT is not reducible to anonymity in the semi-honest model.

In contrast to the situation in the semi-honest model, anonymity is strictly stronger than secure channels in the malicious model. Indeed, we have shown how to base secure channels on anonymity (with $t < N$), whereas anonymity cannot be based on secure channels even with $t \geq N/3$ (due to the impossibility of broadcast in the latter setting). Still, the impossibility of reducing OT to anonymity (with $t \geq N/2$) can be extended to the malicious model as well.

Claim 3.7 *There is no statistically t-secure reduction from N-party OT to anonymity for $t \geq N/2$.*

Proof sketch: Consider an N-party OT functionality, in which player A acts as a sender and B as a receiver. (The remaining $N-2$ players have no inputs or outputs.) Suppose towards a contradiction that there is an unconditionally $\lceil N/2 \rceil$-secure protocol π realizing this OT functionality given oracle access to the anonymity functionality. Now, let $\mathcal{A}$ be a set of $\lceil N/2 \rceil$ players such that $A \in \mathcal{A}$ and $B \notin \mathcal{A}$ and $\mathcal{B} = [N] \setminus \mathcal{A}$. Considering π as a protocol between players in $\mathcal{A}$ and players in $\mathcal{B}$, we get a 2-party 1-private protocol π' for OT over an anonymous 2-player network. The key observation is that anonymity is useless (for the purpose of implementing OT) in the case of two players. More precisely, the 2-party functionality induced by partitioning the players of the n-party anonymity functionality into two sets can be reduced to the secure channel functionality. Since unconditionally secure two-party OT cannot be based on a secure channel alone, the claim follows. □

The proof of Claim 3.7 can be extended to obtain similar negative results for other primitives, such as (N-party versions of) coin-flipping or bit commitment.

The above results imply that the anonymity functionality we defined is *nontrivial* in the sense that it implies key agreement, but on the other hand it is not *complete* for all N-party functionalities with respect to n-secure reductions.

4 Efficiency Improvements Based on Anonymity

In this section we consider different scenarios in which anonymous communication can yield *efficiency* improvements over the best known solutions (even ones that rely on cryptographic assumptions).

[6] Here we are not concerned with the efficiency of the reduction, so we do not attempt to optimize the complexity in terms of ℓ.

4.1 Non-interactive Private Statistics

We show how $n \geq 2$ clients can privately compute statistics (such as mean, standard deviation, correlations) on their combined inputs by each sending few anonymous messages to a central server. Our protocols only requires one-way anonymous communication and are private with respect to an adversary corrupting the server along with an arbitrary number of clients.[7] Note that it is impossible to obtain such non-interactive protocols in the standard model, even if one settles for computational privacy.

Our basic building block is a protocol for integer summation. We assume that each client P_i holds an integer x_i, where $0 \leq x_i < M$. We want to design a protocol in which each client sends a small number of anonymous messages to the server, from which the server can recover the sum of all inputs without learning additional information about the inputs. This basic building block for privately computing the sum immediately allows privacy-preserving computation of the mean of a distributed set of data, and can also be applied to privately compute "suites" of statistics such as: (1) both the mean and variance of a set of numbers, and (2) the means of and covariance between two or more sets of numbers (where each player holds corresponding elements from the sets).[8] The sum protocol can also be used to efficiently compute randomized linear *sketches* of the data that reveal approximate statistics (e.g., an approximate histogram).[9]

For privately computing the sum of values, a simple approach that comes to mind is to let each client P_i send x_i messages to the server (each containing an identical default value), and let the server output the total number of messages it received. This protocol provides perfect privacy, but is prohibitively inefficient when the inputs x_i are large. An additional disadvantage of this simple approach is that it does not support private addition over finite groups, which is particularly useful for computing randomized sketches of data.

Our goal is to obtain a (statistically) private protocol in which the communication complexity is essentially optimal: the total number of bits sent by each player is only polylogarithmic in M.

The protocol SUM**.** We present a protocol for adding n inputs in a finite group G of size L. (The above integer summation problem reduces to addition over $G = Z_L$, where $L = nM$.) To compute the sum of the inputs, each player *additively shares* its input into $k = \ell + \sigma$ shares in G, where $\ell = \lceil \log |G| \rceil$ and σ is a statistical security parameter, and sends each share to $\mathcal{S}$ in a separate anonymous message. The server can recover $\sum x_i$ by adding up (in G) the kn messages it received.

Analysis. We now analyze the parameters for which mixing additive shares hides the values of the shared secrets.[10] We start with the case of $n = 2$ players and consider the experiment of running the above protocol with uniformly chosen inputs in G. Let (X, Y) denote the players' random inputs and V denote the mixed shares received by $\mathcal{S}$. Let $V(x, y)$ denote the distribution of V conditioned on $X = x, Y = y$ and $V(x)$ denote the distribution of V conditioned on $X = x$. Finally, let U be a random variable uniformly distributed in G, independently of V.

Lemma 4.1 *Suppose* $\log \binom{2k}{k} > \ell + \sigma$. *Then,* $SD((V, X), (V, U)) \leq 2^{-\Omega(\sigma)}$.

[7] We provide no guarantee of correctness in the presence of malicious clients. However, in most applications of the kind considered here malicious clients can cause nearly as much damage also in an idealized implementation involving a trusted party.

[8] It is important that a suite of statistics are being computed – for instance, in example (1) above, we cannot use the sum protocol to privately compute *only* the variance, without revealing the mean. However, it is most often desirable to compute both the mean and variance together.

[9] In general, the approximate output together with the randomness used to generate the sketch may reveal a few bits of additional information that do not follow from the exact output (see [20]). However, in most applications of sketching this privacy loss is either insignificant or non-existent.

[10] A simpler variant of the problem we consider here was implicitly considered in the context of constructing pseudorandom generators based on subset sum [30].

Proof: For $a \in G^{2k}$ and $\pi \in \binom{[2k]}{k}$ let $h_a(\pi) = \sum_{i \in \pi} a_i$. Note that h_a defines a family of pairwise independent hash functions from $\binom{[2k]}{k}$ to G. (Pairwise independence follows from the fact that each a_i is an independent element of the group.) Also note that (V, X) is distributed identically to $(V, h_V(\Pi))$, where Π is the uniform distribution over all sets in $\binom{[2k]}{k}$ independently of V. This follows from the fact that, by symmetry, every possible k-subset of shares is equally likely to coincide with the shares of X. Finally, the Leftover Hash Lemma (see Lemma 2.1) guarantees that $SD((V, h_V(\Pi)), (V, U)) \leq 2^{-\Omega(H_\infty(\Pi)-\ell)} = 2^{-\Omega(\log \binom{2k}{k}-\ell)}$ from which the lemma follows. □

Lemma 4.2 *Suppose $SD((V, X), (V, U)) \leq \epsilon$. Then for all $x, y, x', y' \in G$ such that $x + y = x' + y'$ we have*

$$SD(V(x, y), V(x', y')) \leq 2|G|^2 \cdot \epsilon.$$

Proof: If $SD((V, X), (V, U)) \leq \epsilon$ then, by Markov's inequality, for every $x \in G$ we have

$$SD(V(x), V) \leq |G| \cdot \epsilon.$$

By the triangle inequality, for every x, x' we have $SD(V(x), V(x')) \leq 2\epsilon|G|$.

To complete the proof we show that a δ-distinguisher D between $V(x, y)$ and $V(x', y')$, where $x + y = x' + y'$, can be turned into a $\delta/|G|$-distinguisher D' between $V(x)$ and $V(x')$. Such a distinguisher can be implemented as follows. Let $z = x + y (= x' + y')$. Given a challenge v (a vector of $2k$ mixed shares), D' checks whether the shares add up to z and if so invokes D on v; otherwise it outputs 0. □

From these two lemmas, we immediately conclude that the protocol privately computes the sum for $n = 2$ players, with the appropriate setting of k:

Theorem 4.3 *Let $k = 1.5\ell + \sigma$. Then protocol* SUM *privately computes the sum of $n = 2$ inputs in a group G, where $|G| < 2^\ell$, with statistical error $2^{-\Omega(\sigma)}$.*

This analysis also implies private computation for $n > 2$ as well:

Theorem 4.4 *Let $k = 1.5\ell + \sigma + \log n$. Then protocol* SUM *privately computes the sum of n inputs in a group G, where $|G| < 2^\ell$, with statistical error $2^{-\Omega(\sigma)}$.*

Proof: (sketch) First we note that if the adversary corrupts q out of the n players and learns their secrets, then, since all other shares remain independent and uniformly mixed, the problem reduces to the case of an adversary that has made no corruptions among $n - q$ players. Therefore, without loss of generality, we may assume that the adversary knows none of the secrets of the n players (but of course he knows the sum).

Let $x, y \in G^n$ be two distinct sets of player inputs such that $\sum x_i = \sum y_i$. We will argue that the adversary cannot distinguish (statistically, to within $2^{-\Omega(\sigma)}$ error) between its view when x defines the inputs, or when y defines the inputs.

We define a "basic step from x" to be a vector x' such that there exist two indices $i, j \in [1, n]$ and a value $a \in G$, such that $x'_i = x_i + a$ and $x'_j = x_i - a$. Then, it is easy to see that to reach y from x requires at most $n - 1$ basic steps. Thus, by a standard hybrid argument, we need only show that the adversary cannot distinguish between x and y, when y is a basic step from x. Let the indices i and j be fixed to reflect this basic step.

We now argue that if there exists an adversary A that can distinguish its view based on x from its view based on y, then there is an adversary A' for the $n = 2$ case that can distinguish its view when the secrets are (x_i, x_j) from its view based on (y_i, y_j). The reduction is simple: Since x and y agree on all other coordinates except i and j, the adversary A' (which will have all other coordinate values of x built into it) can generate shares for all $x_u (= y_u)$ for $u \neq i, j$, and mix these shares uniformly into the $2k$ shares it obtains as input, in

order to perfectly simulate the views of the adversary A when given either x or y. Thus, if A succeeds, then so does A' with precisely the same probability of success. But since we know by the previous theorem that no such A' can exist, we conclude that no such A can exist, and the theorem is established. The parameter values needed follow naturally from this argument. □

The above analysis (specifically, the proof of Lemma 4.1) does not require perfect anonymity. Rather, we consider the adversary's uncertainty about which of the shares it sees could be shares of any particular secret; because we are in the statistical case of unbounded adversaries, we can model this as a distribution over sets (a distribution Π over $\binom{[2k]}{k}$). As long as this distribution has enough min-entropy, our argument applies. Thus, the level of anonymity needed is only as much as needed to guarantee high min-entropy in the adversary's uncertainty distribution. In particular, the protocol remains secure even if the server can *partially* correlate messages sent from the same client, e.g., by grouping messages according to their exact time of arrival.

4.2 Secure Two-Party Computation

From here on we consider the following common scenario. There are n clients who communicate with a server $\mathcal{S}$ via two-way anonymous channels. The server holds an input x and each client i an input q_i. The goal is for each client to learn the value $f(q_i, x)$ for some function f, while keeping q_i private from the server and (possibly) prevent clients from learning additional information about x. We refer to these two privacy properties as *client privacy* and *server privacy*, respectively.

A first observation is that we cannot generally hope to obtain unconditional privacy against arbitrary collusions of parties. Indeed, this would contradict the impossibility of OT discussed in Section 3.4. In the following we show how to utilize the private summation protocol for obtaining full client privacy together with weak server privacy that holds with respect to every single, semi-honest client.

Linear functions. Suppose that $f(q, x) = x \cdot q$, where x is a matrix held by the server and q a vector held by the client. In this case, we can obtain a protocol with full client privacy and weak server privacy as follows. (1) each client breaks his input vector q to k additive shares $q_1, \ldots, q_k$ and anonymously sends k messages of the form (j, q_j) to $\mathcal{S}$; (2) $\mathcal{S}$ replies to each message of the form (j, q_j) with $X \cdot q_j + r_j$, where the r_j are random masks that add to 0. Each client can now recover its output by adding up the k messages it received. Client privacy follows by a similar analysis to the sum protocol, letting $k = O(|q| + \log n + \sigma)$ (where σ is a statistical security parameter). The only difference is that now the shares obtained by the servers are labeled by indices j. This decreases the entropy of the server's uncertainty by a small factor that can be easily compensated by a larger choice of k.

General functions. Using known reductions, it is possible to reduce any secure two-party computation task to a matrix-vector product of the form $f(q, x) = x \cdot q$, where q is determined by the client's input and x is a *randomly* chosen based on the server's input. This reduction can efficiently support information-theoretic server privacy for functions with small formulas or branching programs (e.g., using [32]), and computational server privacy for general functions (e.g., using [54]).

4.3 Private Information Retrieval

The above general protocols support a weak notion of server privacy. Requiring no server privacy whatsoever, any function f as above can be trivially computed by letting the server send x to each client. However, such solutions are often unsatisfactory in cases where x is very large and the function f has a low communication complexity. A canonic problem of this sort is that of Private Information Retrieval (PIR) [15],

defined by the functionality $f(i, x) = x_i$; that is, the PIR functionality allows each client to privately retrieve a selected item i from a large database x. (PIR is also useful as a building block for a large class of low-communication private protocols [44].) We denote the number of records in the database x by m and assume by default that each record consists of a single bit.

In the following, we use anonymous channels to obtain a PIR protocol which allows a server to handle queries that may originate from many different clients using nearly optimal communication and computation. In contrast to previous protocols presented in this work, the current protocol only provides *computational* client privacy. Its privacy relies on a natural generalization of a previous intractability assumption from [42, 43] related to noisy polynomial reconstruction.

The model. We consider a system with a single server and several (typically many) clients. The protocol will require only a single round of queries and answers. Each client can send several (simultaneous) queries to the server and receive a separate answer for each query. Thus, the interaction between the clients and the server is captured by a single invocation of the two-way anonymity functionality defined in Appendix A (generalized to allow several queries from each client). Our protocol will provide the following, somewhat unconventional, security guarantee. An adversary corrupting the server and a subset of the clients will be unable to learn the inputs of the remaining clients, in the sense that different choices for these inputs induce computationally indistinguishable views, provided that the number of *uncorrupted* clients exceeds some given threshold. (More precisely, it will suffice that the total number of queries originating from uncorrupted clients exceed this threshold.) The value of the threshold will depend on the database size and the security parameter, but not on the number of clients. Thus, the fraction of corrupted parties that can be securely tolerated by the protocol tends to 1 as the number of clients grows. We note that while we assume for simplicity that all clients send their queries simultaneously in a synchronous way, this requirement is not essential. As in other protocols presented in this work, the security of the PIR protocol relies on having sufficient uncertainty (from the adversary's point of view) about the origin of queries sent by uncorrupted clients. To guarantee privacy even when there is only a small number of active clients, one can employ "dummy clients" that generate and send sufficiently many random queries to the server. We stress that in our protocol the clients do not need to interact with each other, nor even be aware of the number of other clients in the system.

Overview of construction. The high level idea is the following. We take a t-server information-theoretic PIR protocol in which the client's privacy is protected against each collusion of k servers. (In our typical choice of parameters, we let k serve as the security parameter and $t = O(k \cdot m^{\epsilon})$.) The t queries sent by the client in this protocol can be viewed as points on a degree-k curve in a low-dimensional space. Any k of these t points jointly reveal nothing about the client's selection i, whereas any $k + 1$ of them completely determine i. A natural approach that comes to mind is to (computationally) hide the curve encoding i by adding random noise. As it turns out, the required amount of noise is very large – it has to be at least of the order of magnitude of m, the database size, in order to defeat a recent attack by Coppersmith and Sudan [17].[11] Thus, the approach is entirely useless in case of a single client accessing the database. The key observation is that the same amount of noise would suffice to hide an arbitrarily large number of curves, possibly originating from different clients. Thus, the use of anonymity allows to amortize the required noise over multiple clients. As long as the number of uncorrupted clients is sufficiently large (say, comparable to the database size) the amount of noise each client needs to contribute is small.

The original polynomial reconstruction (PR) intractability assumption, introduced by Naor and Pinkas in [42, 43] (see also [35]), asserts the following. For an appropriate choice of parameters, the output of

[11] An attempt to base PIR on a stronger version of our intractability assumption was made in [34]. This assumption is broken by the Coppersmith-Sudan algorithm (see also [8]). Our protocol requires a much more conservative choice of parameters.

the following experiment keeps a secret field element $s \in F$ semantically secure with respect to a security parameter k: (1) pick a random polynomial $p(\cdot)$ of degree$\leq k$ such that $p(0) = s$; (2) pick t distinct evaluation points $a_1, \ldots, a_t \in F$ and n random noise coordinates $r_1, \ldots, r_n \in F$; (3) output the good points $(a_j, p(a_j))$ along with noise points (r_j, b_j) in a random order (where each b_j is random and independent of all r_i).[12]

The Guruswami-Sudan list decoding algorithm [28] implies that the above assumption does not hold when $t > \sqrt{(n+t)k}$. Thus, the assumption becomes plausible only when the amount of noise is higher, say when $n \gg t^2/k$. We rely on the following multi-dimensional variant of the above assumption: the secret s is replaced by a vector of c field elements $s = (s_1, \ldots, s_c)$ and the polynomial p by a $(c+1)$-dimensional curve, namely by a vector of c polynomials $p = (p_1, \ldots, p_c)$. The above experiment can then be generalized in a natural way to the multi-dimensional case. Formally, the assumption is defined as follows.

Definition 4.5 (Noisy Curve Reconstruction (CR) Assumption) *Let k denote a degree parameter, which will also serve as a security parameter. Given functions $F(k)$ (field), $c(k)$ (dimension), $t(k)$ (points on curve), and $n(k)$ (noise), we say that the* CR assumption *holds with parameters (F, c, t, n) if the output of the following experiment keeps a secret $s \in F(k)^{c(k)}$ semantically secure (with respect to security parameter k):*

- *Pick random polynomials $p_h(\cdot)$, $1 \leq h \leq c$, such that each p_h is of degree$\leq k$ and*

$$p(0) \stackrel{def}{=} (p_1(0), \ldots, p_c(0)) = s;$$

- *Pick t distinct evaluation points $a_1, \ldots, a_t \in F \setminus \{0\}$ and n random noise coordinates $r_1, \ldots, r_n \in F \setminus \{0\}$;*
- *Output the good points $p(a_j)$ along with random noise points $(b_j^1, \ldots, b_j^c) \in_R F^c$ in a random order.*

In the following, it is convenient to consider an augmented (and "more adversarial") experiment which outputs the evaluation point a_j along with each c-tuple $p(a_j)$ and a random element of F along with each noise point b_j. Clearly, if the augmented CR assumption (i.e., the CR assumption with respect to the augmented experiment) holds then it also holds with respect to the original experiment. The algorithm from [17] breaks the augmented CR assumption when $t > ((n+t)k^c)^{1/(c+1)} + k + 1$. Thus, when $t = o(nk^c)^{1/(c+1)}$ or equivalently $n = \omega(t \cdot (t/k)^c)$ the augmented assumption (let alone the original one) remains plausible.

Our protocol uses the following choice of parameters. Let $c > 1$ be a constant. (The amortized complexity per client will be of the order of $n^{1/c}$.) We view the entries of the database x as the coefficients of a c-variate polynomial q_x of total degree at most $d = O(m^{1/c})$ over a field F, where the size of F will be specified later. This allows to associate with each selection index $i \in [m]$ a point $z_i \in F^c$ such that $q_x(z_i) = x_i$ (see, e.g., [15]).

The protocol. Each client, holding selection index i, picks a random degree-k curve $p = (p_1, \ldots, p_c)$ such that $p(0) = z_i$, as well as $t = kd + 1$ random distinct evaluation points $a_j \in F \setminus \{0\}$. It anonymously sends to the server the t queries v_j where $v_j = p(a_j) \in F^c$. In addition, the client anonymously sends a number of random noise points of the form $b_j \in_R F^c$, so that the total number of noise points sent by all clients is at

[12] The corresponding assumption in Definition 2.3 of [43] differs in that it requires the noise points r_j to be distinct from the good points b_j. Our variant of the assumption is slightly stronger, since the choice of points reveals a small amount of information about the locations of the points a_j (to an extent which diminishes with the field size). This information can be eliminated by picking the points a_j in a completely independent way (i.e., with repetition), replacing multiple occurrences of a good point a_j with noise points. We prefer the above variant because it simplifies the formulation of our protocol.

least n. (The security of the protocol will be guaranteed as long as the total number of noise points sent by *uncorrupted* clients is at least n.) The server replies to each query with an answer $s_j = q_x(v_j)$. (If all values of q_x were precomputed, this is done via a table lookup.) Finally, the client can recover x_i by interpolating the degree-kd univariate polynomial defined by the points (a_j, s_j). For this interpolation to be possible, we need $|F| > t + 1$, though a larger F is desirable for enhancing the security.[13]

Privacy. The following lemma guarantees that if n points of noise are sufficient to (computationally) hide the selection of a single client, then this is also the case for an arbitrary polynomial number of clients.

Lemma 4.6 *Let k denote a security parameter let $u(k)$ be a polynomial. Let $A(k) = (A_1(k), \ldots, A_{u(k)}(k))$ be a distribution ensemble, where $A(k)$ is a sequence of $u(k)$* independent *distributions over* multisets *of elements from a domain $D(k)$. Let $B(k) = (B_1(k), \ldots, B_{u(k)}(k))$ be another distribution ensemble as above, and let $R(k)$ be a random multiset of $n(k)$ elements from $D(k)$. Moreover, suppose that for every index sequence $j(k)$, $1 \leq j(k) \leq u(k)$, we have $A_j \cup R \stackrel{c}{\approx} B_j \cup R$. (Here $\stackrel{c}{\approx}$ denotes computational indistinguishability with respect to polynomial-size circuits, and the dependence of all parameters on k is implicit in the notation.) Then, $A_1 \cup \ldots \cup A_u \cup R \stackrel{c}{\approx} B_1 \cup \ldots \cup B_u \cup R$.*

Proof: Suppose the contrary. By a hybrid argument, there is a sequence $j(k)$ such that $A_1 \cup \ldots \cup A_{j-1} \cup B_j \cup \ldots \cup B_u \cup R$ can be distinguished from $A_1 \cup \ldots \cup A_j \cup B_{j+1} \cup \ldots \cup B_u \cup R$ with non-negligible advantage. The corresponding distinguisher T can be used to obtain a distinguisher between $A_j \cup R$ and $B_j \cup R$: given a multiset S, take the union of S with a sample from $A_1 \cup \ldots \cup A_{j-1} \cup B_{j+1} \cup \ldots \cup B_u$, and invoke T on the resulting multiset. □

Note that the CR assumption guarantees that the queries of a *single* client, when combined with n points of noise, keep the client's selection computationally private. Thus, Lemma 4.6 establishes the privacy of the protocol for an arbitrary number of clients, with the same amount of noise as that required for the privacy of a single client:

Theorem 4.7 *If the CR assumption (Definition 4.5) holds with parameters $(F(k), c(k), t(k), n(k))$, then the above anonymous PIR protocol remains computationally private for an arbitrary (polynomial) number of clients, as long as the total amount of noise contributed by uncorrupted clients is at least $n(k)$.*

Parameters. Recall that we set c to be a constant and let $t = O(km^{1/c})$. As noted above, a good choice of the noise parameter for the CR assumption is $n = \omega(t \cdot (t/k)^c) = \omega(k \cdot m^{1+1/c})$. Thus, the total amount of noise is indeed comparable to the database size. Finally, we need to argue that the query space is polynomial and can therefore be computed by the server. Recall that we require that $|F| = \Omega(t) = \Omega(km^{1/c})$ but, as discussed above, it is safer to avoid many collisions and thus let $|F|$ be larger than n. Either way, the size of the query space $|F|^c$ is polynomial in m.

Using the above choice of parameters, we get:

Corollary 4.8 *Let $m(k)$ be the size of the database as a function of the security parameter. Let c be a positive integer and $\epsilon > 0$ a constant such that the CR assumption holds with parameters $(F(k), c, t(k), n(k))$, where $t = O(k \cdot m^{1/c})$, $n = O(k \cdot m^{1+1/c+\epsilon})$, and $|F(k)| = O(k \cdot m^{1/c+\epsilon})$. (A larger value of ϵ represents a more conservative assumption.) Then, assuming two-way anonymous communication, there is a one-round PIR protocol involving a single server and multiple clients in which the amortized communication and computation per query are $\tilde{O}(t) = \tilde{O}(km^{1/c})$. The protocol remains computationally private as long as uncorrupted clients make* together *at least $n(k)$ random noise queries.*

[13] The problem with letting $|F| \approx t$ is that the good points are likely to share the same X-coordinates with many noise points. In such a case, PR-type assumptions are susceptible to lattice-based attacks [9].

4.4 "PIR to Peer"

The feasible query domain of the above protocol allows to distribute the role of the server between many users without compromising efficiency or security. This gives rise to the following, conceptually attractive, type of distributed storage systems.

We envision a peer-to-peer community in which a large number of users are willing to share their resources. In such a community, each user may play three distinct roles: a *database owner*, holding some data to which it wishes to provide private access; a *server*, making its small share of contribution for each database owner in the system; and a *client*, wishing to privately retrieve data from other database owners.

The efficiency of such systems is measured by three main parameters. A first parameter is the *communication complexity* required for retrieving an item from a database, which we require to be sublinear in the size of the database. Note that achieving this in our setting implies that only a small fraction of the servers should be involved in each PIR invocation. A second efficiency parameter is the (expected) total *load* on the servers for each query made by a client. We would like the load of answering the clients' queries to be distributed evenly between the servers, even if there is a "popular" item requested by many clients. Finally, we would like the storage overhead to be small, and evenly distributed between the servers. In terms of *security*, we would like to ensure that (assuming that the network provides a reasonable level of anonymity) the privacy of each query made by the client is protected even when almost all users are corrupted. Finally, we would like the system to be *robust* in the sense that it maintains its functionality even if a large number of users are adversarially corrupted.

To the end of implementing such a system we distribute the role of the single server in the anonymity-based PIR protocol described above. To store a database in the system, the (preprocessed) string containing the list of all possible PIR answers for this database is broken evenly between many different users, acting as servers. (Ideally, the number of users is sufficiently large so that each user receives at most a single element of F for each database in the system.) To access the ith entry in a database x, a user first computes a set of (randomized) queries as in the above PIR protocol, and then fetches the answers by *anonymously* contacting the users that hold the answers to these queries. This scheme has the same (nearly optimal) efficiency and privacy features as in the single-server setting, except that here we additionally get the following *load balancing* feature: the load of answering the queries is evenly distributed between users regardless of the multiset of queries being asked. In particular, even if all users in the system try to access the same data item, the (expected) load on each user remains the same. This feature is due to the randomness of the PIR queries. The randomness of the PIR queries also makes the system *robust* to denial-of-service attacks involving a large fraction of the users. We note that without the use of anonymity the system would still enjoy most of the above features, except that privacy would only hold against small collusions of users (rather than against collusions involving "almost all" users).

Acknowledgements. We thank Andreas Pfitzmann for pointing out the relevance of [2, 47] to implementing key agreement based on anonymity, and Matthias Fitzi for pointing out the relevance of [48] to implementing broadcast based on anonymity. We also thank David Chaum, Juan Garay, Venkat Guruswami, and Tatsuaki Okamoto for helpful discussions and pointers.

References

[1] Anonymity bibliography. `http://www.freehaven.net/anonbib/`

[2] B. Alpern and F. B. Schneider. Key exchange Using 'Keyless Cryptography'. *Information Processing Letters* Vol. 16, pp. 79-81, 1983.

[3] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval. In *Proc. 43rd FOCS*, pages 261–270, 2002.

[4] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*. Springer-Verlag, 20–24 August 2000.

[5] A. Beimel and T. Malkin. A Quantitative Approach to Reductions in Secure Computation. In *Proc. of 1st TCC*, pages 238-257, 2004.

[6] C. H. Bennett, G. Brassard, and J. M. Robert. Privacy Amplification by Public Discussion. *SIAM J. Comput.* 17(2): 210-229 (1988).

[7] R. Berman, A. Fiat, and A. Ta-Shma. Provable Unlinkability against Traffic Analysis. In *Proc. of 8th Financial Cryptography*, Vol. 3110, pp. 266-280, 2004.

[8] D. Bleichenbacher, A. Kiayias, and M. Yung. Decoding of Interleaved Reed Solomon Codes over Noisy Data. *Proc. of ICALP 2003*, pages 97-108.

[9] D. Bleichenbacher and P. Q. Nguyen. Noisy Polynomial Interpolation and Noisy Chinese Remaindering. *Proc. of EUROCRYPT 2000*. pp. 53-69.

[10] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Proc. EUROCRYPT '99*, pages 402–414.

[11] R. Canetti. Security and composition of multiparty cryptographic protocols. In *J. of Cryptology*, 13(1), 2000.

[12] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, Vol. 24(2), pp. 84-88, 1981. Also: UC Berkeley M.Sc. Thesis, 1979.

[13] David Chaum. Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA. In *Proc. EUROCRYPT 1988*, pages 177-182.

[14] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward Privacy in Public Databases. To appear in *Proc. of 2nd TCC*, 2005.

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

[16] B. Chor and E. Kushilevitz. A Zero-One Law for Boolean Privacy (extended abstract). In *Proc. STOC 1989*, pages 62-72.

[17] D. Coppersmith and M. Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proc. of 35th STOC*, pp. 136-142, 2003.

[18] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *Proc. EUROCRYPT* 1999, pages 311-326.

[19] I. Dinur and K. Nissim. Revealing information while preserving privacy. *Proc. of 22nd PODS*, pp. 202-210, 2003.

[20] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. Secure Multiparty Computation of Approximations. In *Proc. ICALP 2001*.

[21] M. J. Fischer and R. N. Wright. Multiparty Secret Key Exchange Using a Random Deal of Cards. CRYPTO 1991: 141-155.

[22] M. Fitzi, J. Garay, U. Maurer, and R. Ostrovsky. Minimal Complete Primitives for Secure Multi-party Computation. In *Proc. Crypto 2001*, pp. 80-100.

[23] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *Proc. STOC 2000*, pp. 494-503.

[24] M. Fitzi, S. Wolf, and J. Wullschleger. Pseudo-signatures, Broadcast, and Multi-party Computation from Correlated Randomness. *Proc. CRYPTO 2004*, pages 562-578.

[25] C. Gentry and Z. Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. *Proc. ICALP 2005*, pages 803-815.

[26] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[27] V. Guruswami and A. Rudra. Explicit Capacity-Achieving List-Decodable Codes. ECCC Report 133, 2005. To appear in STOC 2006.

[28] V. Guruswami, and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, Vol. 45(6), pp. 1757-1767, 1999. Early version: Proc. of 39th FOCS, pp. 28-39, 1998.

[29] D. Harnik, M. Naor, O. Reingold, and A. Rosen. Completeness in two-party secure computation: a computational view. *Proc. STOC 2004*, pages 252-261.

[30] R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *Proc. of 30th FOCS*, pp. 236-241, 1989. Journal version: J. Cryptology 9(4), pp. 199-216, 1996.

[31] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. *Proc. of 30th FOCS*, pp. 248-253, 1989.

[32] Y. Ishai and E. Kushilevitz. Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. *Proc. of ICALP '02*.

[33] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, 13–15 June 2004.

[34] A. Kiayias and M. Yung. Secure Games with Polynomial Expressions. In *Proc. of 28th ICALP*, LNCS 2076, pp. 939-950, 2001.

[35] A. Kiayias and M. Yung. Cryptographic Hardness based on the Decoding of Reed-Solomon Codes with Applications. ECCC Technical Report #017, 2002.

[36] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th STOC*, pages 20–31, 1988.

[37] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky: Reducibility and Completeness in Private Computations. SIAM J. Comput. 29(4): 1189-1208 (2000).

[38] E. Kushelevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proc. of the 38th FOCS*, page 364-373, 1997.

[39] H. Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. *Proc. ISC 2005*, pages 314-328. Full version on eprint.

[40] U. M. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory* 39(3): 733-742, 1993.

[41] T. Moran and M. Naor. Basing Cryptographic Protocols on Tamper-Evident Seals. *Proc. of ICALP 2005*, pp. 285-297.

[42] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. *Proc. of 31st STOC*, pp. 245–254, 1999.

[43] M. Naor and B. Pinkas. Oblivious polynomial evaluation. Manuscript. Available at `http://www.wisdom.weizmann.ac.il/~naor/onpub.html`.

[44] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. *Proc. of STOC 2001*, pp. 590-599.

[45] N. Nisan and D. Zuckerman. Randomness is Linear in Space. *J. Comput. Syst. Sci.*, Vol. 52(1), pp. 43-52, 1996.

[46] F. Parvaresh and A. Vardy. Correcting Errors Beyond the Guruswami-Sudan Radius in Polynomial Time. *Proc. of FOCS 2005*, pp. 285-294.

[47] A. Pfitzmann and M. Waidner. Networks without user observability – design options. In *Proc. Eurocrypt '85*, pages 245-253, 1986. Revision in: *Computers and Security* 6/2 (1987) 158-166.

[48] B. Pfitzmann and M. Waidner. Information-Theoretic Pseudosignatures and Byzantine Agreement for $t \geq n/3$. IBM Research Report RZ 2882 (#90830), IBM Research Division, Zurich, 1996.

[49] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proc. 21st STOC*, pages 73–85. ACM, 1989.

[50] J. F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. *Workshop on Design Issues in Anonymity and Unobservability*, LNCS 2009, pp. 10-29, 2001.

[51] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur.i*, Vol. 1(1), pp. 66-92, 1998.

[52] Daniel R. Simon. Anonymous Communication and Anonymous Cash. *Proc. of CRYPTO 1996*, pp. 61-73.

[53] P. L. Vora. Information Theory and the Security of Binary Data Perturbation. In *Proceedings of Indocrypt 2004*.

[54] A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pp. 162–167, 1986.

A Network Model Definitions

In this section we include few formal definitions needed in Section 2.1.

We start with defining the functionality Anon (that we are going to invoke repeatedly). A first attempt to define this functionality is as follows:

- INPUT: each player P_i provides the functionality with a pair (m_i, d_i), where either m_i is a message (string) and $d_i \in [N]$ is its destination,[14] or $m_i = d_i = \perp$ (indicating that P_i has no message to send).
- OUTPUT: each player P_j outputs the *multiset* of all values m_i for which $d_i = j$.
  The adversary's output includes, for *every* j, the multiset of messages received by player P_j (i.e., the adversary learns the content of all messages at each receiver but does not get any information as for the sender of each message).

(Note that since the output of a player P_j is just a multiset then, in particular, this output reveals no information about the identities of the senders of these message.)

The above formulation of the functionality Anon is not satisfying as it does not allow to deal with a *rushing* adversary; as such, this makes the primitive of anonymous channels too powerful. We modify the definition so as to allow the adversary to see first the content of all messages sent by players which are not under its control (excluding, of course, the identity of the sender of each of these messages) and only then to decide on its own messages. The definition of Anon is therefore described as a two-stage process, as follows:

- INPUT ROUND 1: each player P_i provides the functionality with a pair (m_i, d_i), where either m_i is a message (string) and $d_i \in [N]$ is its destination, or $m_i = d_i = \perp$ (indicating that P_i has no message to send or that P_i is under the adversary's control).
- OUTPUT ROUND 1: No player has an output; the adversary's output includes, for every j, the multiset of messages sent to player P_j.
- INPUT ROUND 2: No player has an input; the adversary's input includes a set of pairs of the form (m, d), where m is a message and d is its destination; the number of these pairs is bounded by the number of players under the adversary's control.
- OUTPUT ROUND 2: each player P_j outputs the *multiset* of all values m_i for which $d_i = j$ (inserted as inputs in any of the two input rounds).

We will also consider a "private" version of Anon, denoted PrivAnon. This functionality is defined similarly to Anon, except that the adversary only learns the contents of the messages sent to corrupted players, rather than the contents of all messages sent in the network.

Next, we define two-way anonymous communication. As before, this can be formalized as an interactive functionality as follows.

- INPUT ROUND 1: each player P_i provides the functionality with a pair (m_i, d_i), as above.
- OUTPUT ROUND 1: each player P_j outputs the *set* S_j of all values m_i for which $d_i = j$.
- INPUT ROUND 2: each player P_j (after performing some local computation) provides the functionality with a vector of values m'_i, one value for each $m_i \in S_j$.

[14] Note that we allow, and in fact in certain cases also use, the possibility that $d_i = i$; i.e., a player may send an anonymous message to itself (clearly, this is useful only for confusing the adversary regarding the messages sent in the network).

- OUTPUT ROUND 2: each player P_i, for which $m_i \neq \perp$, outputs the message m'_i, corresponding to the message m_i that P_i sent to P_j (in ROUND 1).

In the above definition we ignore, for simplicity, the issue of rushing. We do so because the applications in which we use *two-way* anonymous channels are not sensitive to the issue of rushing; in any case, the definition can be easily modified to deal with rushing in the same way as we did above for the one-way case.

B An alternative Key-Agreement protocol

We note that an alternative means for obtaining key-agreement protocols in our setting is via a reduction to the problem of key agreement using a "deck of cards" [21] (see also [40]). In this setting, there is a deck of cards, where a set of cards C_1 is given to player A, another (disjoint) set of cards C_2 is given to player B, and the remaining cards are revealed to the adversary. Fischer and Wright show how to take advantage of the mutual information between A and B so as to agree on a secret key. Below we sketch how to use their protocol in a setting where A, B have access to a public anonymous channel (i.e., this protocol does *not* require private anonymity). The reduction proceeds as follows. A and B both send to B, using the public anonymous channel, ℓ random "cards" from an exponentially large domain. (Malicious players can possibly inject their own uncalled-for values.) Then, B sends back to A the set of $m \geq 2\ell$ received values, eliminating possible duplicates. The received values define a "deck of cards" from which each of A and B knows ℓ (random) cards and the adversary knows the rest. Now, A and B can agree on a key by using the methods of [21].

C Reducing Broadcast to Anonymity

In this section we present a statistically t-secure reduction from broadcast to PrivAnon for $t < n/2$. To this end, it suffices to realize a secure 3-cast protocol, where the 3-cast functionality is the broadcast functionality restricted to some set of 3 players. Using the results of Fitzi and Maurer [23], this implies broadcast with resilience $t < n/2$.

The 3-cast protocol involves a sender S and two receivers A and B. Sender S has an input bit b. In the end of the protocol, both A and B should output the same value, which is equal to b if S is honest. Moreover, the adversary cannot learn b unless it corrupts one of these three players.

The protocol proceeds as follows:

1. A (and B) repeats the following k times: generate (X_0, X_1) (and (Y_0, Y_1) respectively) where each X_b (and Y_b, respectively) is randomly chosen from $\{0,1\}^k$, and A and B send these ordered pairs to S via a private anonymous channel. Thus, S receives k messages of the form $((X_0, X_1), (Y_0, Y_1))$, but S does not know whether the X-tuple came from A or B.

2. S then sends to each player $2k$ values of the form X_b or Y_b, chosen from each ordered pair according to S's input bit b. This communication is done over authenticated non-private channels from S to both A and B.

3. A and B exchange all information they got from S over authenticated channels. A generates a list L_A consisting of all X values received directly from S as well as through B, and B does the same to generate its list L_B. (If both A and B are honest, then $L_A = L_B$.) A counts the number of X_1 values that A generated from among the list L_A, to obtain the quantity m_A. Similarly, B obtains m_B by counting the Y_1 values.

4. A and B exchange m_A and m_B over authenticated channels.

5. A flips a fair coin α. A defines the threshold $T = k/3$ if $\alpha = 0$, and $T = 2k/3$ if $\alpha = 1$. Then A informs B of the threshold value T using an authenticated channel.

6. A now proceeds as follows. If $m_A < T$ and $m_B < T$, then it outputs 0. If both $m_A > T$ and $m_B > T$, then it outputs 1. Otherwise, it repeats this entire protocol at most k times. B does the same. (Note each party only outputs once. It continues to participate in future trials when desired by the other player, but the later trials do not affect its output.)

7. If by the end of all k trials, one of the two players has still not decided its output, then A outputs 0 iff $m_A \leq k/2$, and/or B outputs 0 iff $m_B \leq k/2$.

The correctness of the above protocol is argued as follows.

- First, note that if S, A, B are all honest, then $m_A = m_B$ and is either equal to 0 or k.

- If A is dishonest and S and B are honest we need to prove that B's output will be b with overwhelming probability, no matter what A does.

  However, the only way that A can affect B's output is if it manages to give to B values of the form Y_{1-b} that B chose. However, this would contradict the privacy of the anonymous channel used by B to send its values to S. Hence, if $b = 0$, then $m_B = 0$ and if $b = 1$ then $m_B = k$ in step 3. Hence B will halt with the correct value. A symmetric argument holds when B is dishonest.

- Now consider the case when S is dishonest, and A and B are honest. We need to prove that both A and B get the same output with overwhelming probability. If A and B disagree, that means that they ended up on 2 different sides of the threshold in Step 6. Now, we claim that m_A and m_B can differ by an $\Omega(k)$ additive amount with exponentially small probability. This is because of the following:

  First, note that because A and B are both honest and share the information they receive from S, the only thing that matters is the union of the messages sent to A and B by S. Furthermore we condition on the case that all X and Y values are distinct, since this happens with overwhelming probability.

  For each tuple of the form $((X_0, X_1), (Y_0, Y_1))$ that S received, S has a choice to either:

  1. Send both X_1 and Y_1. In this case both m_A and m_B increase.
  2. Send neither X_1 nor Y_1. In this case both m_A and m_B remain unchanged.
  3. Send only one of X_1 or Y_1. Since by the anonymity property, S does not know which came from A and which came from B, in this case S has precisely $1/2$ probability of increasing m_A by 1 and leaving m_B unchanged, and $1/2$ probability of achieving the opposite.

  In order for S to cause A and B to disagree, S must create an $\Omega(k)$ difference between m_A and m_B. Since the only way to do this is for S to cause situation (3) above $\Omega(k)$ times, a Chernoff bound shows that the probability that S succeeds is negligible.

  Therefore, with overwhelming probability A and B will agree.