Cryptography from Anonymity *

Yuval Ishai[†] Eyal Kushilevitz[‡] Rafail Ostrovsky[§] Amit Sahai[¶]

November 15, 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 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, and hence general multi-party protocols with unconditional security in the presence of an honest majority. In contrast, anonymity cannot be generally used to obtain unconditional security when there is no honest majority.
- **Efficiency.** We show that anonymous channels can yield substantial 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 with respect to *both* communication and computation. A key observation that underlies these results is that *local randomization* of inputs, via secret-sharing, when combined with the *global mixing* of the shares, provided by anonymity, allows to carry out useful computations on the inputs while keeping the inputs private.

1 Introduction

There are many scenarios in which anonymous communication can be implemented at a 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 (see [1, 12, 51, 7] and references therein). Anonymizing web browsers and anonymous email accounts are already

*An extended abstract of this paper appears in [36].

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

[‡]Computer Science Department, Technion. Research supported by grant 36/03 from the Israel Science Foundation and grant 2002354 from the U.S.-Israel Binational Science Foundation. E-mail: eyalk@cs.technion.ac.il

[§]Computer Science Department and Department of Mathematics, UCLA. Supported in part by IBM Faculty Award, Intel equipment grant, NSF Cybertrust grant No. 0430254, Xerox Award and grant 2002354 from the U.S.-Israel Binational Science Foundation. E-mail: rafail@cs.ucla.edu

[¶]Computer Science Department, UCLA. Research supported in part by an Alfred P. Sloan Foundation Research Fellowship, an Intel equipment grant, NSF ITR/Cybertrust grants 0456717 and 0627781, and grant 2004361 from the U.S.-Israel Binational Science Foundation. Email: sahai@cs.ucla.edu

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 additionally hiding *what* actions are being performed. In the context of distributed computation, anonymity allows hiding which users hold which local inputs, whereas privacy requires hiding 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?

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 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 the "cryptography from anonymity" paradigm in two settings: (1) Establishing *feasibility* results for traditional cryptographic tasks unconditionally, based solely on the assumption of public 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 using 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. 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 senders, except when the sender itself is corrupted. In

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

addition to such anonymous channels, we also assume that the players can communicate via authenticated but public point-to-point channels, and (in some cases) a public broadcast channel[2].

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. *Private* anonymous channels, allowing the adversary to learn only messages sent or received by corrupted players. This construction can tolerate an arbitrary number of malicious players assuming the availability of a broadcast channel.

2. Secure point-to-point channels with unconditional security against an arbitrary number of malicious players. (This protocol strengthens the simple key agreement protocol described above in that it is resilient against "Denial of Service" attacks mounted by malicious players.) This result does not require a broadcast channel.

3. General multi-party protocols with unconditional security against any minority of malicious players. This result assumes the existence of a broadcast channel in addition to the public anonymous channel, and follows from [50] and the result above.

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, the above 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 do not restrict 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 us to keep the inputs private and, at the same time, allows us 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

[2]We note that the conference version of this paper [36] suggested that public anonymous channels implied broadcast. In fact, the protocol given in [36] for constructing private anonymous channels from public anonymous channels needs broadcast channels. We fix this by making an explicit use of broadcast as a primitive.

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*. (Indeed, in our analysis it will be useful to view the local randomization as a seed to a randomness extractor, and the partial randomization provided by anonymity as an imperfect source.) More concretely, our approach is to first have 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 carrying out the desired computation 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 is 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 two or more clients hold m-bit integers and 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 well with the bit length m. Towards a more efficient solution, we use the split and mix approach in the following natural 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, 41] 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 [41, 10, 42, 26] 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 full 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, 35]. However, all previous solutions to this problem either require multiple servers (and fail to protect against colluding servers) or only allow to amortize the cost of several simultaneous queries that *originate from the*

same client.[3] A remaining open problem in 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. We suggest a solution to this problem using (two-way) anonymous communication, and assuming that the queries are made simultaneously by the 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 *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 precomputing the answers to all possible sub-queries.

The security of our PIR protocol relies on an intractability assumption related to the hardness of reconstructing *noisy* low-degree curves in a low-dimensional space. A similar assumption for the two-dimensional case was introduced by Naor and Pinkas [45]. Roughly speaking, the original assumption from [45] asserts that noisy two-dimensional curves cannot be reconstructed in a much better way than using the Guruswami-Sudan list decoding algorithm [30]. Our generalized assumption asserts that, in the low-dimensional case, one cannot do much better than the Coppersmith-Sudan algorithm [17] (which, in a sense, extends [30] to the multi-dimensional case). We note that this assumption does not seem to be affected by the recent progress in the field of list-decoding [47, 29, 28]. It is also instructive to note that this assumption (as well as the one from [45]) is not known to imply public-key encryption, let alone PIR, in the standard setting. Accordingly, in the basic version of our protocol the *total* communication with all clients must be larger than the database size (yet the amortized cost per client becomes low when the number of clients grows). We show how to get around this limitation by combining our basic protocol with a standard single-server PIR protocol. The resulting protocol is close to optimal with respect to both communication and computation even when the number of clients is smaller than the database size.

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. Most of our protocols only require "sufficient uncertainty" about the origins of messages to maintain their full security, at a modest additional cost. 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. See Appendix A for further discussion.

1.2 Related Work

A variant of the toy example presented above (for key agreement using anonymity) was suggested by Alpern and Schneider [2]. A similar idea was previously used by Winkler [55] for establishing secure channels in

[3] 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.

the game of Bridge. Our work, in contrast, achieves 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 [48]. Pfitzmann and Waidner [49] use a variant of private anonymous communication as an intermediate step for obtaining highly resilient broadcast protocols. Finally, Anonymous communication has also been exploited in the context of practically oriented applications such as voting [13] and electronic cash [53].

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 [39, 40, 23, 31, 5, 25, 44] 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.

Finally, our approach is also reminiscent of work on data privacy through data *perturbation* (cf. [19, 14, 54]). 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.

Organization. In Section 2, we provide some necessary background and definitions. In Section 3, we apply anonymity for obtaining new feasibility results and, in Section 4, we apply anonymity for improving the efficiency of cryptographic protocols.

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 [33], asserting that a pairwise independent hash function can be used as a strong randomness extractor [46].

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 a *non-private* point-to-point authenticated channel. (By "non-private" we mean that the adversary learns all messages sent over the channels.) 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.[4] 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 how to construct the latter, stronger primitive from the former. 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 appears in Appendix B. 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, 52]). As before, this can be formalized as an interactive functionality, as described in Appendix B.

It is easy to implement two-way anonymous channels from one-way anonymous channels and a public broadcast channel: each sender sends its message m to the desired receiver (using one-way anonymous channel); each receiver broadcasts 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 applications 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

[4] 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.

have access to a trusted oracle computing f). For a formal definition of "statistically t-secure protocols", see [11, 27]. For simplicity we consider *non-adaptive* adversaries, who choose the set of corrupted players in advance.

3 Feasibility Results Using Anonymity

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

3.1 From Public to Private Anonymity

We start by showing how to realize the private anonymity functionality PrivAnon using the basic (non-private) anonymity functionality Anon, together with broadcast.

The above goal is nontrivial even if we were additionally given private (non-anonymous) point-to-point channels. Indeed, there is no obvious way to combine the advantages of non-private but anonymous channels and private 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. We wish to construct a protocol allowing A and B to send messages to R with the following properties: (1) If A and B are honest then their anonymity is preserved. (2) If the adversary corrupts only one sender, then it cannot violate the privacy or the correct delivery of the message sent by the other sender, nor can it correlate its own message with the other message.

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 by using a one-time pad for encrypting and an unconditionally-secure MAC for authenticating. We let k denote a statistical security parameter. The protocol proceeds as follows:

1. Repeat $2k$ times sequentially (each iteration is referred to as a "session"):

   (a) Each of A, B and R sends a random k-bit number to R, using anonymous (non-private) channels. (With overwhelming probability, the honest players choose distinct numbers. The adversary may send its own numbers and, being rushing, may duplicate numbers sent by honest players.)

   (b) R considers the numbers received in the previous step, ignoring repetitions, and chooses 2 out of these numbers, including its own. R sends these 2 numbers in lexicographic order to both players A and B via the authenticated broadcast channel. The order of these 2 numbers defines a (secret) bit between R and either A or B according to who's number is chosen.

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 honest sender 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 the key corresponding to the message).

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 and the use of the broadcast channel.

If the receiver R and one sender, A, are honest, but the other sender B is dishonest, we must guarantee privacy, integrity, and independence 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 an N-player 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. The resulting protocol is a statistically N-secure reduction from PrivAnon to Anon.

3.2 Secure Point-to-Point Channels

In this section, we describe a variant of the protocol from the previous section. This protocol uses public anonymous channels to realize secure point-to-point communication. Recall that the simple key agreement protocol described 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 as well as the other users in the system are honest-but-curious.

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, possibly malicious, players. The main difficulty in utilizing anonymity in this case is that when one of the players needs to send an anonymous message, 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.

1. Repeat $2Nk$ times sequentially (each iteration is referred to as a "session"):

(a) Each of A, B sends a random k-bit number to B, using anonymous (non-private) channels. (With overwhelming probability, the honest players choose distinct numbers. The adversary may send its own numbers and, being rushing, may duplicate numbers sent by honest players.)

(b) B considers the numbers received in the previous step, ignoring repetitions, and chooses two out of these numbers, including its own. B sends these two numbers in lexicographic order to A via an authenticated (but non-private) point-to-point channel.

(c) A reports to B, over an authenticated channel, whether its number was included in the pair sent by B. In such a case, the order of the two numbers defines a secret bit between A and B.

2. If there are at least k successful sessions, the first k such sessions define a shared k-bit key between A and B which both players output. Otherwise, A and B output independently random k-bit strings. [If both A and B are honest, the latter event will occur with negligible probability.]

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

Proof sketch: We only need to guarantee security of the protocol when both A and B are honest. In this case, we can condition on the event that A and B always generate distinct k-bit numbers in every session, since this event will happen with all but negligible probability. We need to argue that in such a case: (1) A and B will eventually output the same k-bit key (except with negligible probability), and (2) that this key is hidden from the adversary.

(1) follows from the fact that in each session the adversary can send at most $N-2$ messages that interfere with the two messages sent by A and B, and thus the success probability of each session is at least $1/N$.

(2) follows from the fact that in each successful session, the adversary learns no information about the bit shared by A and B due to the anonymity of the channel. ■

In Appendix C 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 [22] (see also [43]).

3.3 General Secure Multiparty Computation

We turn to the question of basing general secure multiparty computation (MPC) on anonymity. Combined with the implementation of private point-to-point channels from the previous section, one can then apply known MPC protocols (e.g., [50] or [18]) and obtain general t-secure MPC with $t < N/2$ when given a broadcast channel.

Theorem 3.4 *For any N-party functionality f, there is a statistically t-secure reduction of f to* Anon *and broadcast for any $t < N/2$.*

We now argue that the bound $t < N/2$ in Theorem 3.4 is tight.

Claim 3.5 *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 3.5 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 allows 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 Using 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).

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 require one-way anonymous communication and are private with respect to an adversary corrupting the server along with an arbitrary number of clients.[5] Note that it is impossible to obtain such non-interactive protocols in the standard model, even if one settles for computational privacy. (It is possible to solve this problem in the non-interactive model of [20]; however, such a solution requires setup assumptions and provides a weaker security guarantee.)

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).[6] The sum protocol can also be used to efficiently compute

[5] 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.

[6] 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

randomized linear *sketches* of the data that reveal approximate statistics (e.g., an approximate histogram).[7]

As discussed in the Introduction, a simple solution to the integer summation problem 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 depends only logarithmically on 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 shares in G (where k will be specified later) 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.[8] 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,* $SD((V, X), (V, U)) \leq 2^{-\Omega(\sigma)}$.

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 [33] (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

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

and variance together.

[7] 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 [21]). However, in most applications of sketching, this privacy loss is either insignificant or non-existent.

[8] A somewhat simpler variant of the problem we consider here was implicitly considered in the context of constructing pseudorandom generators based on subset sum [32].

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 $2^{-\Omega(\sigma)}$.*

The following theorem extends the analysis to the case of $n > 2$ clients.

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: 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.

Note that our analysis requires the amount of communication per client to grow with the number of clients. This might seem counter-intuitive, since a larger number of clients should intuitively make it easier to "blend in the crowd". Recall, however, that we require security to hold against an adversary who may corrupt the server and *an arbitrary number of clients*. Considering an adversary who corrupts the server and all but two clients, the communication per client in the case of n clients cannot be smaller than in the case of just two clients. We also note that the additive $\log n$ term in the complexity is not just an artifact of the analysis; it can be shown that the communication involving each client has to grow to infinity with the number of clients.

4.2 Secure Two-Party Computation

In Appendix D we describe an application of the summation protocol for realizing general secure two-party computation between the server and each client, albeit in a rather weak security model.

4.3 Private Information Retrieval

In this section, we use anonymous channels to obtain a PIR protocol which allows a server to handle queries that may originate from many different clients using a nearly optimal amount of 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 [45] related to noisy polynomial reconstruction.

The model. We consider a system with a single server, holding a database $x \in \{0,1\}^m$, and several (typically many) clients. Each client holds a selection index $i \in [m]$ and wishes to learn x_i without revealing i to the server. The protocol requires only a single round of queries and answers. Each client can send several (simultaneous) anonymous 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 B (generalized to allow several queries from each client).[9] We stress that in our protocol the clients do not need to interact with each other. 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 exceeds 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.

[9] 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.

Overview of construction. We take a t-server information-theoretic PIR protocol in which the client's privacy is protected against collusions 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 an attack by Coppersmith and Sudan [17].[10] 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. When the number of uncorrupted clients is sufficiently large, the amount of noise each client needs to contribute is small.

The original polynomial reconstruction (PR) intractability assumption, introduced by Naor and Pinkas [45] (see also [38]), asserts roughly the following. For an appropriate choice of parameters, the output of 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).[11]

The Guruswami-Sudan list decoding algorithm [30] 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 $p_1(\cdot), \ldots, p_c(\cdot)$, s.t. each p_h is of degree $\leq k$ and*

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

[10] An attempt to base PIR on a stronger version of our intractability assumption was made in [37]. This assumption was broken by the Coppersmith-Sudan algorithm (see also [8]). Our protocol relies on a much more conservative choice of parameters, that is not known to imply PIR in the standard model.

[11] The corresponding assumption in Definition 2.3 of [45] 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.

- *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.*

Towards relating the CR assumption to known attacks, 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. We stress that the assumption does not seem to be affected by the recent progress in the field of list-decoding [47, 29, 28].

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 a database $x \in \{0,1\}^m$ 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 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.[12]

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

[12] 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].

advantage. The corresponding distinguisher T can be used to get 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 result. ■

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 $t = O(km^{1/c})$. 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 comparable to the database size. Finally, we argue that the query space is polynomial, so that the answers to all queries can be precomputed 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 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 is computationally private as long as uncorrupted clients make* together *at least $n(k)$ random noise queries.*

Achieving sublinear communication. While the PIR protocol given by Corollary 4.8 has a low complexity per client when the number of clients is large, it requires the total communication with all clients to be bigger than the database size. Indeed, a protocol with a sublinear total communication would imply a PIR protocol and hence key agreement in the standard model, which is not known to be implied by the CR assumption. We now briefly sketch a way for combining the protocol $\mathcal{P}$ of Corollary 4.8 with any standard (single-server) PIR protocol $\mathcal{P}'$ in order to reduce the communication complexity when the number of clients $u(k)$ is smaller than the database size $m(k)$.

Let $m'(k)$ be a database size for which $\mathcal{P}$ remains secure if each of the u clients contributes a single noise query. (Note that $m'(k)$ should always be smaller than $u(k)$, to an extent that depends on the strength of the CR assumption; when c is big and ϵ is small, $m'(k)$ is close to $u(k)$.) We parse the m bits of the original database x as an $m' \times \ell$ matrix X where $\ell = \lceil m/m' \rceil$. Each client, who wishes to retrieve entry (i, j) of X, invokes the protocol $\mathcal{P}$ as if it is retrieving the i-th bit from a database of size m'. Along with each of the sub-queries in $\mathcal{P}$, it sends a (standard) PIR query pointing to the j-th entry of a database of ℓ entries, generated according to $\mathcal{P}'$. (An independent invocation of $\mathcal{P}'$ is used for each sub-query.) For

each sub-query received from a client, the server obtains ℓ answers, each resulting from applying $\mathcal{P}$ to the corresponding column of X, and then computes a single response by applying $\mathcal{P}'$ to the database of ℓ answers. The resulting protocol has a low communication complexity if so do $\mathcal{P}$ and $\mathcal{P}'$. Excluding the cost of pre-processing, the amount of server computation per client is typically of the order of ℓ, resulting in a total amount of computation that is close to $\ell \cdot m' = m$ when m' is close to u. Thus, a good choice of parameters yields a protocol which is close to optimal with respect to both communication and computation, regardless of the number of clients.

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, 48] to implementing key agreement based on anonymity, and Matthias Fitzi for pointing out the relevance of [49]. We also thank David Chaum, Juan Garay, Venkat Guruswami, Tatsuaki Okamoto, Farzad Parvaresh and the anonymous referees 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, pages 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. *Journal of Cryptology*, 17(2), pages 125–151, 2004. Earlier version in CRYPTO 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*, pages 266-280, 2004.

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

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

[10] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Proc. of 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), pages 84-88, 1981. Also: UC Berkeley M.Sc. Thesis, 1979.

[13] D. 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. In *Proc. of 2nd TCC*, pages 363–385, 2005.

[15] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *J. of the ACM*, 45:965–981, 1998. Earlier version in FOCS '95.

[16] B. Chor and E. Kushilevitz. A Zero-One Law for Boolean Privacy. *SIAM J. Discrete Math* 4(1): 36-47, 1991. Earlier version in STOC '89.

[17] D. Coppersmith and M. Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proc. of 35th STOC*, pages 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. In *Proc. of 22nd PODS*, pp. 202-210, 2003.

[20] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *Proc. of 26th STOC*, pages 554-563, 1994.

[21] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. Secure Multiparty Computation of Approximations. In *Proc. 28th ICALP*, pages 927-938, 2001.

[22] M. J. Fischer and R. N. Wright. Multiparty Secret Key Exchange Using a Random Deal of Cards. In *Proc. CRYPTO 1991*, pages 141-155.

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

[24] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *Proc. 32nd STOC*, pages 494-503, 2000.

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

[26] C. Gentry and Z. Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In *Proc. 32nd ICALP*, pages 803-815, 2005.

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

[28] V. Guruswami and F. Parvaresh. Personal communication.

[29] V. Guruswami and A. Rudra. Explicit Capacity-Achieving List-Decodable Codes. In *Proc. 38th STOC*, pp. 1-10, 2006.

[30] V. Guruswami, and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, Vol. 45(6), pages 1757-1767, 1999. Earlier version in FOCS '98.

[31] D. Harnik, M. Naor, O. Reingold, and A. Rosen. Completeness in two-party secure computation: a computational view. In *Proc. 36th STOC*, pages 252-261, 2004.

[32] R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *J. Cryptology* 9(4), pages 199-216, 1996. Earlier version in FOCS '89.

[33] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. In *Proc. 30th FOCS*, pages 248-253, 1989.

[34] Y. Ishai and E. Kushilevitz. Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. In *Proc. 29th ICALP*, pages 244-256, 2002.

[35] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *Proc. 36th STOC*, pages 373-382, 2004.

[36] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from Anonymity. In *Proc. FOCS 2006*.

[37] A. Kiayias and M. Yung. Secure Games with Polynomial Expressions. In *Proc. 28th ICALP*, pages 939-950, 2001.

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

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

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

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

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

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

[44] T. Moran and M. Naor. Basing cryptographic protocols on tamper-evident Seals. In *Proc. of 32nd ICALP*, pages 285-297, 2005.

[45] M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.* 35(5), pages 1254-1281, 2006. Earlier version in STOC '99.

[46] N. Nisan and D. Zuckerman. Randomness is Linear in Space. *J. Comput. Syst. Sci.*, Vol. 52(1), pages 43-52, 1996. Earlier version in STOC '93.

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

[48] 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.

[49] 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.

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

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

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

[53] D. R. Simon. Anonymous Communication and Anonymous Cash. In *Proc. CRYPTO 1996*, pages 61-73.

[54] P. L. Vora. Information Theory and the Security of Binary Data Perturbation. In *Proc. Indocrypt 2004*, pages 136-147.

[55] P. Winkler. Cryptologic techniques in bidding and defense: Parts I, II, III, and IV. *Bridge Magazine*, April–July 1981.

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

A On Using Imperfect Anonymity

While we assume for simplicity that the network provides *perfect* anonymity, this assumption can be relaxed. In fact, most of 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 in the Introduction. 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. We argue that 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.

B Anonymity Models

We now present some formal definitions of the anonymity functionalities described in Section 2. We start by defining the functionality Anon that captures our basic model of non-private anonymous communication. 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,[13] 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

[13] 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).

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 $m_i \in S_j$.
- 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.

C An alternative Key-Agreement protocol

An alternative means for obtaining key-agreement protocols using anonymity is via a reduction to the problem of key agreement using a "deck of cards" [22] (see also [43]). In this problem, 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 [22].

D Secure Two-Party Computation

In this section as we consider the following 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 preventing 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 3.3. Thus, we will settle for the following relaxed notion of privacy. First, the server's privacy will only be guaranteed against a single, *semi-honest* client. As to the privacy of the clients, we will have the following guarantee: the statistical advantage of a (potentially malicious) adversary corrupting the server and a subset of the clients is exponentially small in the number of uncorrupted clients. Thus, our security model is best suited to scenarios in which protecting the privacy of the clients is the main concern.

Note that, unlike the non-interactive protocol for secure summation, here we inherently need to have a two-way interaction between the clients and the server. Thus, in this setting it is possible to obtain (computational) security without anonymity at all, assuming the existence of oblivious transfer. The advantage of using anonymity is in obtaining a light-weight protocol that avoid the need for public-key operations. Our protocol can either obtain unconditional security (for restricted function classes) or computational security based on the existence of one-way functions (for arbitrary functions).

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. For this case, consider the following protocol. (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. Note that by the choice of the masks r_j, each individual client will learn no information about the output. As to the clients of the privacy, here we can prove the following property: with an appropriate choice of k (see below), any collusion of the server with a subset of the clients will only be able to learn the *sum* of the inputs of the uncorrupted clients. (We will later show how to eliminate this extra information when there are many uncorrupted clients.) This 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 labelled by indices j. This decreases the entropy of the server's uncertainty by a small (logarithmic) 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 matrix 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 [34]), and computational server privacy for general functions (e.g., using [56]). Combined with the above protocol for linear functions, this gives us the desired result, except for the fact that the *sum* of the client's (transformed) inputs is now revealed. We eliminate this extra information by using additional (local) randomization of both the clients' inputs and the server's input.

We illustrate this using the simple case of the AND function. This would give the general result, as securely computing this function can be used *in parallel* for computing arbitrary functions using randomization techniques from [56, 34]. In the case of the AND function, the server holds a bit x. Each client i holds a bit q_i and would like to learn x AND q_i (namely learn x only if $q_i = 1$). Applying the above protocol for linear functions over GF(2) (where x is viewed as a 1×1 matrix), we get a protocol in which the server can learn the exclusive-or of all inputs. One simple approach for eliminating this extra bit of information is by assuming that among the uncorrupted clients there is at least one "helper" client who picks its input at random. Note, however, that we cannot let a client holding an input q_i serve also as a helper client, since such a client will be able to recover x from the $2k$ answers even if $q_i = 0$.

Instead, we use the following randomization approach. The server breaks his secret bit x into 3 shares (x_1, x_2, x_3) such that a single share gives no information about x and any pair of shares completely reveals x. Each client holding an input 0 will transform its input to a random vector from the set $\{(0,0,0), (1,0,0), (0,1,0), (0,0,1)\}$. Similarly, a client holding an input 1 will transform its input to a random vector from the set $\{(1,1,1), (1,1,0), (1,0,1), (0,1,1)\}$. Now the clients and the server engage in parallel invocations of the basic AND protocol, resulting in each client learning the shares x_i corresponding to the positions in which its chosen triple contains 1. Note that each client will be able to recover x if its input is 1 and will learn no information about x otherwise, as required. The server, on the other hand, will learn the exclusive-or of all triples held by the clients. It can be shown using a Markov chain analysis that this exclusive-or converges exponentially fast to the uniform distribution over $\{0,1\}^3$. Thus, the server's advantage vanishes exponentially with the number of uncorrupted clients. Note that the above choice of triples is not completely arbitrary; for instance, eliminating $(0,0,0)$ from the first set would make the server's view in the case all clients hold 0 statistically far from the case all clients hold 1.