J. Cryptology (2001) 14: 37–74
DOI: 10.1007/s001450010008

Journal of
CRYPTOLOGY

Universal Service-Providers for Private Information Retrieval[*]

Giovanni Di Crescenzo
Telcordia Technologies, 445 South Street,
Morristown, NJ 07960-6438, U.S.A.
giovanni@research.telcordia.com

Yuval Ishai
Department of Computer Science, Technion,
Haifa 32000, Israel
yuvali@cs.technion.ac.il

Rafail Ostrovsky
Telcordia Technologies, 445 South Street,
Morristown, NJ 07960-6438, U.S.A.
rafail@research.telcordia.com

Communicated by Oded Goldreich

Received 21 September 1998 and revised 21 December 1999
Online publication 9 May 2000

Abstract. A *private information retrieval* scheme allows a user to retrieve a data item of his choice from a remote database (or several copies of a database) while hiding from the database owner which particular data item he is interested in. We consider the question of private information retrieval in the so-called "commodity-based" model, recently proposed by Beaver for practically oriented service-provider Internet applications. We present simple and modular schemes allowing us to reduce dramatically the overall communication involving users, and substantially reduce their computation, using off-line messages sent from service-providers to databases and users. The service-providers do not need to know the database contents nor the future user's requests; all they need to know is an upper bound on the data size. Our solutions can be made resilient against collusions of databases with more than a majority (in fact, all-but-one) of the service-providers.

Key words. Private information retrieval, Secure computation, Commodity-based cryptography, Communication complexity.

[*] An extended abstract of this work appeared in *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC* '98. Part of this work was done while Yuval Ishai was visiting Telcordia Technologies.

1. Introduction

Cryptography in the 90*s.* With the widespread use of World-Wide Web and Internet applications, cryptographic protocols are increasingly used in commercial settings. Hence, while the trend of the 1980s was to establish general plausibility results, the trend of the 1990s was to consider solutions which are *both* provably secure and efficiently implementable in practical applications (for a more general discussion on this topic see surveys by Goldreich [15] and Goldwasser [19], [18]). This is the case of the current work as well—here we show how with the help of service-providers one can maintain the user's privacy while retrieving information from a remote database with almost the same total communication cost to/from the user as if we do not care about privacy at all.

Commodity-Based Cryptography. Motivated by a client–server paradigm, Beaver [2] proposed a new "commodity-based" approach for the design of efficient cryptographic protocols. In his model there are several independent service-providers, called *commodity servers* (or simply *servers* for short), which off-line sell "security commodities" to their clients; these commodities can be later utilized by the clients to perform more cheaply various cryptographic tasks. An advantage of this model is that the servers do not need to know private inputs of their clients, do not need to know which or how many other servers are being used, and only send a single message (commodity) to each client. On the other hand, this setting is clearly much more restrictive than the usual setting for secure multiparty protocols (as in [31], [17], [6], and [8]), which allows point-to-point multiround communication. In [2] Beaver showed how to achieve so-called "1-out-of-2 Oblivious Transfer" and "multicast" in this model, provided that the majority of the servers are honest. We consider this model in the context of remote database information retrieval.

Private Information Retrieval. *Private information retrieval* (*PIR*) schemes allow a user to retrieve a data item of his choice from a remote database while hiding which particular data item he is interested in. In the basic PIR setting the database content is modeled by an n-bit string x, possibly replicated (for a reason that will be explained shortly) in $k \geq 1$ distinct databases. The user, holding a *retrieval index* i, wishes to learn the ith data bit x_i. A t*-private PIR scheme* is a protocol between the user and the databases in which the user learns x_i while keeping i private from any collusion of t databases (where the user's privacy is either information-theoretic or computational, depending on the setting). By default, PIR refers to 1-private PIR.

A trivial single-database solution to this problem is to let the database send its entire content x to the user; however, while being information-theoretically private, the *communication complexity* of this solution may be prohibitively large. (For example, consider retrieval from a Web search-engine.) While it is impossible to do better using a single database and maintaining information-theoretic user privacy [10], it turns out that if x is replicated in two or more databases, then there are much better solutions. We now briefly mention some of the work done in this area in the past. Private information retrieval with information-theoretic user privacy was introduced by Chor et al. [10], who constructed (1-private) schemes with a communication complexity of $O(n^{1/3})$ bits for $k = 2$ databases, $O(n^{1/k})$ bits for a constant number $k \geq 3$ of databases, and $O(\log^2 n \log\log n)$ bits for $k = O(\log n)$ databases. Ambainis [1] improved the k-database upper bound to $O(n^{1/(2k-1)})$ for any constant k (see [22] for an improved dependence on k). Generalizations to t-private PIR were given in [10] and [22].

Computational PIR schemes, in which the user's privacy should only hold with respect to computationally bounded databases (relying on certain intractability assumptions), were first considered by Chor and Gilboa [9], who constructed a 2-database scheme with subpolynomial communication and by Ostrovsky and Shoup [28] who considered private reading and writing to/from multiple databases.

Kushilevitz and Ostrovsky [23] constructed the first *single*-database scheme with subpolynomial communication, thereby demonstrating that in the computational setting data replication can be totally avoided. Subsequent improvements to the communication complexity of their scheme (relying on stronger assumptions) were given by Stern [30] and most recently by Cachin et al. [7], the latter achieving polylogarithmic communication. Single-database schemes were shown to imply the existance of one-way functions [5] and Oblivious Transfer protocols [11]. Moreover, single-database schemes with communication complexity strictly smaller than the database size were recently shown to exist based on any one-way trapdoor permutation [24].

Our Setting. The setting may be informally described as follows. Similarly to the original PIR scenario, there is a user holding a retrieval index i and $k \geq 1$ databases holding copies of an n-bit data string x. As before, the user wishes to retrieve x_i without revealing i to the databases. However, in our setting there are additionally one or more *commodity servers* which may off-line send randomized messages, called commodities, to the user and to each of the databases. A *commodity-based PIR scheme* (or *commodity scheme* for short) consists of the following two stages:

1. (Off-line *commodity distribution stage*.) Each server, on input 1^n and an optional security parameter 1^κ, independently runs a probabilistic polynomial time sampling algorithm, outputting $k + 1$ strings (commodities) sent via secure channels to the user and the k databases.
2. (On-line *retrieval stage*.) With commodities from the off-line stage as private inputs, the user and the databases execute some PIR protocol in which the user sends queries to the databases and receives answers in return.

In a real-life setting we envision many (perhaps competing) servers, where the user decides which and how many of them to use. We measure both the off-line communication in the commodity distribution stage (i.e., size of commodities) and the on-line communication between the user and the databases in the retrieval stage. Our main objectives are to minimize the total communication involving the user (in both stages) and to shift most of the overall communication to the off-line stage. It should be noted that one clearly cannot expect to achieve a better *total* complexity than that of ordinary (i.e., serverless) PIR schemes, since the servers in any commodity scheme can be simulated by the user to obtain an ordinary scheme of the same total complexity. We stress though that all of our schemes allow minimizing the total communication involving the *user* to be logarithmic in n (and polynomial in the security parameter in the single-database case).

Another major goal is to guarantee the user's privacy even when some of the servers dishonestly collude with databases. One less obvious motivation for protecting against such collusions is that a nonmalicious yet faulty server (e.g., one with a bad random number generator) may cause the same damage as a server which colludes with the databases. (In contrast, faulty databases do not compromise the user's privacy, neither in our schemes nor in previous PIR schemes.) In most of our multiserver schemes, even if all but one of the servers collude with databases, the user's privacy still remains intact.

Our Results. We start by constructing single-server commodity schemes, where as long as this server does not collude with the databases the user's privacy is guaranteed. We then show how to *compose* such single-server schemes into multiserver schemes with improved privacy properties. In particular, by establishing general transformations from PIR schemes to commodity schemes (and by "plugging in" appropriate modifications of PIR schemes from [10], [9], and [23]) we obtain the following commodity schemes:

- **Computational single-database case:** For any constant integers $m, d \geq 1$ we construct an m-server, single-database computational scheme, withstanding collusions of the database with up to $m-1$ servers, with user's communication complexity $O(\log n + poly(\kappa))$ (counting both the user's commodity and on-line communication) and server-database commodity complexity $O(\kappa \cdot n^{1/d})$ (where κ is a security parameter and security is based on the Quadratic Residuosity Assumption).
- **Computational multi-database case:** For any constant $m \geq 1$ we construct an m-server, 2^m-database computational scheme, withstanding collusions of a database with up to $m-1$ servers, with user's communication of size $O(\log n)$ and server-database commodity complexity $\kappa \cdot 2^{O(\sqrt{\log n})}$ (relying on the existence of a pseudorandom generator). Schemes of this type are most appealing when the server-privacy threshold is small and the database size is large. However, since the number of databases is perhaps the most important complexity measure, such schemes are obviously useless for all but very small values of m.
- **Information-theoretic multi-database case:** For any constant integers $m, t, d \geq 1$, we construct an m-server, $(mtd + 1)$-database information-theoretically private scheme, withstanding collusions of up to t databases and $m-1$ servers, with user's communication complexity $O(\log n)$ and commodity complexity $O(n^{1/d})$. Schemes of this type are most appealing when the database is moderately sized.

We then proceed to show how to make the amortized cost of our commodity schemes cheaper and how to test commodities:

- **Amortizing commodity cost for multiple queries**: In most of our s-private schemes (i.e., those that can withstand s dishonest servers), by using $m > s + 1$ servers the amortized commodity cost per query can be reduced to $(1/(s+1)) \cdot (m/(m-s))$ times the cost of a single query in the $(s+1)$-server scheme (while maintaining s-privacy).
- **Commodity testing**: We give procedures for verifying the validity of commodities supplied by servers, allowing us to ensure correctness of our schemes even in the presence of faulty or malicious servers. This problem is particularly natural in a setting where some of the (potentially many) servers may be malfunctioning. Moreover, the testing procedure can be carried out off-line, after the distribution of commodities and before the actual retrieval.

We finally discuss two extensions of the original problem; one concerns protecting privacy of the data against a potentially dishonest user (in a sense that the user cannot get more information than the single entry he has "paid for," see [13], [30], and [27]) and another concerns extension of our results to the related problem of *private information storage* [28].

Benefits for PIR. As discussed above, reducing the *communication* cost of PIR serves as the main motivation for introducing commodity schemes. Indeed, commodity schemes constructed in this work require little on-line communication and little *total* communication involving the user; furthermore, their communication is typically unbalanced in a favorable direction: almost all of it is directed from servers to their clients (namely, users and databases) and from databases to their clients (namely, users). However, our transformations of PIR schemes into commodity schemes may also be beneficial for reducing the *computation* cost of PIR. A substantial portion of the user's computation (to an extent depending on the underlying PIR scheme) is shifted to an off-line stage and is carried out by the servers. Even if better single-database PIR schemes are devised[1] this advantage may still justify the use of commodity schemes in the computational, single-database case. Finally, a major disadvantage of single-database commodity schemes over their PIR counterparts is that the user's privacy may be compromised if servers collude with the database. To avoid this, one may use a degenerate form of our single-server construction in which the user simulates the server; while obviously not reducing his total work, this shifts most of the user's computation (and communication) to an off-line stage without compromising his privacy in any way.

Comparison with Related Work. It is instructive to illuminate two points of comparison between this work and Beaver's work [2], which introduces the commodity-based model we use. First, protocols from [2] do not dramatically save on-line communication; the main goals there are to provide a level of resilience which is impossible to achieve in the information-theoretic setting without the aid of the servers, and to remove unnecessary interaction. Second, our solutions achieve resilience to collusions of databases with up to $m-1$ servers, in opposition to an optimal threshold of $\lfloor (m-1)/2 \rfloor$ servers in Beaver's Oblivious-Transfer protocol. This higher privacy threshold is made possible here because of the different setting, which allows either replication of data or computational privacy.

A very different PIR model using auxiliary servers was recently proposed by Gertner et al. [12]. This model differs from Beaver's (and our) model in that it allows servers to interact with the user and the databases. The objective of [12] is also different: it is not to decrease the total on-line work or the user's work, but rather to reduce the amount of unprotected data replication in information-theoretic PIR by allowing a database to "secret-share" its content with several data servers.

[1] Recent single-database PIR schemes [30], [7] fall short of being satisfactorily efficient in two ways. First, their computation cost is very high (for instance, the scheme from [7] requires the database to perform n modular exponentiations over a large modulus). Second, even their communication overhead is quite significant for "realistic" choices of parameters, especially when retrieving multibit records.

Organization. Section 2 contains some notation, as well as formal definitions of the PIR and commodity PIR models. In Section 3 we summarize the complexity parameters of specific PIR schemes which can be utilized for obtaining communication efficient commodity PIR schemes. Section 4 introduces atomic commodity schemes, and Section 5 deals with composing them to improve their privacy properties. In Section 6 we construct multi-database schemes based on the method of low-degree polynomial interpolation. In Section 7 we show that the commodity cost of our schemes can be amortized over multiple queries. Section 8 provides procedures for testing the correctness of commodities distributed by the servers. Section 9 discusses two extensions of the original problem. Finally, the appendices contain a description of some PIR schemes referred to in Section 3, as well as more general commodity testing procedures.

2. Preliminaries

2.1. *General Notation*

By Z_n we denote the additive group of residues modulo n and by $\mathrm{GF}(q)$, where q is a prime power, a finite field of order q. Addition, subtraction, and multiplication operations are sometimes carried over a finite group or field, as implied by the context. By $y \oplus z$ we denote the bitwise exclusive-or of the two binary strings y, z. By $\mathcal{R}$ we denote the set of reals, by $\mathcal{R}^+$ the positive reals, by $\mathcal{N}$ the natural numbers, and by $[k]$ the set $\{1, 2, \ldots, k\}$. By $\log n$ we denote $\lceil \log_2 n \rceil$, and by e_r, $r \in Z_n$, the rth unit vector of length n (starting with $r = 0$). We say that a function $\epsilon\colon \mathcal{N} \to \mathcal{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer κ_c such that $\epsilon(\kappa) < \kappa^{-c}$ for all $\kappa \geq \kappa_c$.

By default, an *algorithm* refers to a probabilistic Turing Machine, and an *efficient algorithm* to a probabilistic polynomial time Turing Machine. We model adversaries by nonuniform families of Boolean circuits. The *size* of a circuit F is the number of gates in F. By $F(y)$, where F is an l-input circuit and y is a string over a finite alphabet, we denote the value of the circuit F applied to the l-bit prefix (or padding) of the binary encoding of y.

Whenever referring to a *random* choice of an element from a finite domain A, the associated distribution is uniform over A, and is independent of all other random choices. We use the following notation for defining probabilistic experiments and algorithms. By $e \overset{R}{\leftarrow} E$ we denote a choice of an element e from a distribution E (or uniformly from a finite set E), and by $e \leftarrow v$ the assignment of the value v to e. By $A(y)$, where A is an algorithm, we denote the output distribution of the algorithm A running on input y, where the probability space is induced by the random coins of A. If A is deterministic, $A(y)$ denotes its output value. By $\mathbf{Pr}[e \overset{R}{\leftarrow} E;\ f \overset{R}{\leftarrow} F; \ldots : p(e, f, \ldots)]$, where $p(\cdot, \cdot, \ldots)$ is a predicate, we denote the probability that $p(e, f, \ldots)$ will be true after the ordered execution of the assignments $e \overset{R}{\leftarrow} E;\ f \overset{R}{\leftarrow} F; \ldots$.

2.2. *Parameters for PIR and Commodity Schemes*

We let k denote the number of *databases*, an instance of which is denoted $\mathcal{DB}_j$, and m denote the number of *commodity servers* (or *servers* for short), an instance of which is denoted $\mathcal{S}_h$. A *data string*, denoted x, is held by all k databases and is unknown to the user and the servers. Instead of only considering the default scenario where a single bit is retrieved, we will occasionally be interested in the more general scenario of retrieving an ℓ-bit *record*. To this end we view the data string x as an n-tuple of length-ℓ records, where $\ell = 1$ by default. The position, also called the *index*, of a data record which the user would like to retrieve is denoted by i, where $i \in Z_n$. Notice that under the above notation the data string is an n-tuple from $(\{0,1\}^\ell)^n$ and the desired data record x_i is a string in $\{0,1\}^\ell$. Finally, in the computational setting κ denotes a security parameter.

2.3. *Definitions*

In the following definitions of PIR and commodity-based PIR schemes we restrict our attention to the default setting of *bit retrieval* (i.e., $\ell = 1$ and $x \in \{0,1\}^n$). The more general case is addressed in Section 2.4.

A *PIR scheme* is a randomized protocol, in which the user sends a *query* to each database and receives an *answer* in return.[2] At the end of the interaction, the user applies some *reconstruction* function to the answers, obtaining the desired data bit x_i. A *commodity-based PIR scheme* (or *commodity scheme* for short) consists of: (1) an off-line *commodity distribution stage*, in which each server sends a (possibly different) randomized string, called *commodity*, to the user and to each database; and (2) an on-line *retrieval stage*, which proceeds similarly to an ordinary PIR scheme except that queries, answers, and reconstruction may also depend on commodities. Since PIR schemes may be viewed as serverless commodity schemes, their definition is derived as a special case of the following "generic" definition.

An m-server k-database commodity scheme $\mathcal{C}$ is defined by a quadruple of efficient algorithms $(\mathbf{com}_\mathcal{C}, \mathbf{que}_\mathcal{C}, \mathbf{ans}_\mathcal{C}, \mathbf{rec}_\mathcal{C})$, where:

- $\mathbf{com}_\mathcal{C}(1^\kappa, 1^n, h)$ is the commodity generation algorithm invoked by each of the m servers; given a security parameter κ, data size n, and server identity h, it outputs randomized commodities $(c_h^u, (c_h^{db_1}, \ldots, c_h^{db_k}))$, where c_h^u is sent by $\mathcal{S}_h$ to the user and each $c_h^{db_j}$ to the corresponding database.
- $\mathbf{que}_\mathcal{C}(1^\kappa, 1^n, i, (c_1^u, \ldots, c_m^u))$ outputs a k-tuple of queries $(q_1, \ldots, q_k)$ generated by the user on security parameter κ, data size n, retrieval index i, and commodities $c_1^u, \ldots, c_m^u$ (where c_h^u is the commodity received from server $\mathcal{S}_h$). If $\mathcal{P}$ is a PIR scheme, we also need $\mathbf{que}_\mathcal{P}$ to output an auxiliary *reconstruction information* string z (possibly containing some trapdoor information required for efficient reconstruction) such that reconstruction can later depend on the answers and z alone, without depending on the index i, the queries generated by $\mathbf{que}_\mathcal{P}$, or the random coins of $\mathbf{que}_\mathcal{P}$. Although taking z to include *all* random coins and inputs of $\mathbf{que}_\mathcal{P}$ will always do, it turns out that a much shorter string z can be used in all currently known PIR schemes, without affecting the computational efficiency of reconstruction.[3]

[2] A more general definition would allow multiple rounds of interaction rather than a single queries–answers round. However, all currently known PIR schemes require only a single round of interaction.

[3] In all information-theoretic schemes known to date [10], [1], [22], as well as in the computational scheme of [9], either no such auxiliary reconstruction information is needed or only i is needed. In known single-database computational schemes [23], [30], [7], z of length κ or $\kappa + polylog(n)$ suffices (in [23], for instance, a trapdoor consisting of the factorization of a κ-bit modulus N is sufficient for efficient reconstruction).

This feature, which is not very useful in the original PIR setting, turns out to be important in our context.

- $\mathbf{ans}_{\mathcal{C}}(j, x, q_j, (c_1^{db_j}, \ldots, c_m^{db_j}))$ outputs the answer of database $\mathcal{DB}_j$, $1 \leq j \leq k$, on the data string x, query q_j, and commodities $c_1^{db_j}, \ldots, c_m^{db_j}$.
- $\mathbf{rec}_{\mathcal{C}}((a_1, \ldots, a_k), (c_1^u, \ldots, c_m^u), z)$ outputs a single bit reconstructed by the user from the answers $a_1, \ldots, a_k$, commodities $c_1^u, \ldots, c_m^u$, and (in the case of PIR) reconstruction information z.

The special case of PIR. A k-database PIR scheme $\mathcal{P}$ is defined as a 0-server k-database commodity scheme. Hence $\mathcal{P}$ may be defined by a triple $(\mathbf{que}_{\mathcal{P}}, \mathbf{ans}_{\mathcal{P}}, \mathbf{rec}_{\mathcal{P}})$, where all commodity-related inputs to these three algorithms are omitted.

Before proceeding to specify the semantic requirements a commodity scheme must obey, two further syntactic remarks are in place.

1. Some inputs to the functions $\mathbf{que}_{\mathcal{C}}, \mathbf{ans}_{\mathcal{C}}, \mathbf{rec}_{\mathcal{C}}$ are omitted when they are not needed. For instance, in most of our constructions all servers play a symmetric role, in which case h will be omitted from the inputs of $\mathbf{com}_{\mathcal{C}}$. We also omit the input 1^κ whenever referring exclusively to information-theoretic schemes (which do not require a security parameter). Finally, note that the parameters κ, n are not given as explicit inputs to $\mathbf{ans}_{\mathcal{P}}$ or $\mathbf{rec}_{\mathcal{P}}$; however, they may be implicitly contained in their inputs (for instance, the data string x determines n and a query q_j may determine κ).
2. For any scheme $\mathcal{C}$ we assume that both $\mathbf{ans}_{\mathcal{C}}$ and $\mathbf{rec}_{\mathcal{C}}$ are deterministic. If $\mathcal{C}$ is strictly a commodity scheme (i.e., with $m \geq 1$), we assume that $\mathbf{que}_{\mathcal{C}}$ is also deterministic, which makes the user deterministic as well.

Any commodity scheme must satisfy both correctness and privacy requirements, defined in the next subsections.

2.3.1. *Correctness*

A commodity scheme is said to be *correct* if, at the end of the retrieval stage, the reconstructed value is always equal to x_i (assuming that all parties are honest). This requirement may be relaxed to allow some small reconstruction error (as in [7]); we use the perfect correctness variant for simplicity.

We write two separate correctness definitions, one for PIR and one for commodity PIR with $m \geq 1$, incorporating the above syntactic remarks.

A k-database PIR scheme $\mathcal{P}$ is correct, if, for any $\kappa, n, x \in \{0,1\}^n$, $i \in Z_n$,

$$\begin{aligned}&\mathbf{Pr}[((q_1, \ldots, q_k), z) \overset{R}{\leftarrow} \mathbf{que}_{\mathcal{P}}(1^\kappa, 1^n, i);\\ &\quad (a_1, \ldots, a_k) \leftarrow (\mathbf{ans}_{\mathcal{P}}(1, x, q_1), \ldots, \mathbf{ans}_{\mathcal{P}}(k, x, q_k)) :\\ &\quad \mathbf{rec}_{\mathcal{P}}((a_1, \ldots, a_k), z) = x_i] = 1,\end{aligned}$$

where the probability is over the random coins of $\mathbf{que}_{\mathcal{P}}$.

An m-server k-database commodity scheme $\mathcal{C}$ is correct if, for any $\kappa, n, x \in \{0,1\}^n$,

$i \in Z_n$,

$$\begin{aligned}
\mathbf{Pr}[&(c_1^u, (c_1^{db_1}, \ldots, c_1^{db_k})) \xleftarrow{\text{R}} \mathbf{com}_{\mathcal{C}}(1^\kappa, 1^n, 1);\\
&\vdots\\
&(c_m^u, (c_m^{db_1}, \ldots, c_m^{db_k})) \xleftarrow{\text{R}} \mathbf{com}_{\mathcal{C}}(1^\kappa, 1^n, m);\\
&(q_1, \ldots, q_k) \leftarrow \mathbf{que}_{\mathcal{C}}(1^\kappa, 1^n, i, (c_1^u, \ldots, c_m^u));\\
&(a_1, \ldots, a_k) \leftarrow (\mathbf{ans}_{\mathcal{C}}(1, x, q_1, (c_1^{db_1}, \ldots, c_m^{db_1})),\\
&\qquad\qquad \ldots, \mathbf{ans}_{\mathcal{C}}(k, x, q_k, (c_1^{db_k}, \ldots, c_m^{db_k}))) :\\
&\mathbf{rec}_{\mathcal{C}}((a_1, \ldots, a_k), (c_1^u, \ldots, c_m^u)) = x_i] = 1,
\end{aligned}$$

where the probability is over the random coins of the m independent invocations of $\mathbf{com}_{\mathcal{C}}$.

2.3.2. *Privacy*

Informally, a commodity scheme is said to be (s, t)*-private* (and a PIR scheme t-private) if i is kept private from any collusion of s (possibly dishonest) servers and t databases.[4] We use nonuniform security definitions for convenience; the security of our constructions extends to the uniform setting as well.

Let $T \subseteq [k]$ be the indices of t corrupt databases and let $S = \{h_1, \ldots, h_s\} \subseteq [m]$ be the indices of s corrupt servers, which may distribute arbitrary commodities. We do not restrict the computation of corrupt servers during the commodity distribution stage; hence it may be assumed without loss of generality that commodities sent by these servers are *determined* by κ, n.[5] We specify the (deterministic) corruption strategy of servers from S by a function $S^*(\cdot, \cdot)$, such that $S^*(1^\kappa, 1^n)$ returns a set $\{(h_1, c_{h_1}), \ldots, (h_s, c_{h_s})\}$ specifying the commodities sent by corrupt servers.[6] We let $V_{\mathcal{C}}^{S^*,T} = (C_{\mathcal{C}}^{\bar{S},T}, Q_{\mathcal{C}}^{S^*,T})$ denote the joint *view* of databases from T, consisting of both *commodities* received from incorrupt servers in $\bar{S} \overset{\text{def}}{=} [m]\backslash S$ (included in the random variable $C_{\mathcal{C}}^{\bar{S},T}$) and on-line *queries* (included in $Q_{\mathcal{C}}^{S^*,T}$). More formally, for any $\kappa, n, i \in Z_n$ and S, S^*, T as above, the random variable $V_{\mathcal{C}}^{S^*,T}(\kappa, n, i) = (C_{\mathcal{C}}^{\bar{S},T}(\kappa, n), Q_{\mathcal{C}}^{S^*,T}(\kappa, n, i))$ is obtained as follows: (1) conduct the probabilistic experiment appearing in the correctness definition above, except that for $l = 1, 2, \ldots, s$ replace the h_lth invocation of $\mathbf{com}_{\mathcal{C}}$ by an assignment from the corresponding entry of S^*; (2) let $C_{\mathcal{C}}^{\bar{S},T}$ include all commodities $c_h^{db_j}$ with $h \in \bar{S}$ and $j \in T$; and (3) let $Q_{\mathcal{C}}^{S^*,T}$ include all queries q_j with $j \in T$.[7] Finally, for any (fixed) κ, n, we let $\mathcal{C}(\kappa, n)$ denote a restriction of $\mathcal{C}$ to these specific parameters. Thus, $V_{\mathcal{C}(\kappa,n)}^{S^*,T}(i)$ and $C_{\mathcal{C}(\kappa,n)}^{\bar{S},T}$ are different names for the random variables $V_{\mathcal{C}}^{S^*,T}(\kappa, n, i)$ and $C_{\mathcal{C}}^{\bar{S},T}(\kappa, n)$, respectively.

[4] Dishonest databases do not pose any risk to the user's privacy in our single-round setting.

[5] This follows from a standard "averaging argument"; namely, there is some fixed choice of the dishonest servers' coins given which the adversary's advantage is maintained.

[6] Commodities sent by corrupt servers to *databases* are irrelevant to the user's privacy.

[7] Note that $Q_{\mathcal{C}}^{S^*,T}$ depends on the corruption strategy S^*, as the user's queries depend on his commodities.

Information-Theoretic Privacy. We say that the scheme $\mathcal{C}$ is *information-theoretically* (s, t)*-private* (and refer to it as an information-theoretic scheme) if, for any number of records n, retrieval indices $i_1, i_2 \in Z_n$, collusion $S \subseteq [m]$ of s servers with corruption strategy S^*, and collusion $T \subseteq [k]$ of t databases, the random variables $V^{S^*,T}_{\mathcal{C}(n)}(i_1)$ and $V^{S^*,T}_{\mathcal{C}(n)}(i_2)$ are *identically* distributed.

Computational Privacy. In the computational setting, the above perfect privacy requirement is relaxed to computational indistinguishability, parameterized by the security parameter κ and the data size n.[8] Formally, let $\mathcal{F}$ be a class of functions $\mathbf{f}$: $\mathcal{N} \times \mathcal{N} \to \mathcal{N}$, specifying a bound on the adversary's resources (as a function of κ, n), and let $\mathcal{E}$ be a class of functions ε: $\mathcal{N} \times \mathcal{N} \to \mathcal{R}^+$, specifying a bound on the tolerated advantage of an adversary in distinguishing between different retrieval indices.

For any two distributions D_1, D_2, circuit F, and constant $\epsilon > 0$, we say that F *distinguishes between D_1 and D_2 with an ϵ-advantage* if $|\mathbf{Pr}[F(D_1) = 1] - \mathbf{Pr}[F(D_2) = 1]| \geq \epsilon$. For any $S \subseteq [m]$,$T \subseteq [k]$, $\epsilon > 0$, and positive integers f, κ, n, we say that *the collusion (S, T) can (f, ϵ)-break $\mathcal{C}(\kappa, n)$*, if there exists a corruption strategy S^* for servers in S, retrieval indices $i_1, i_2 \in Z_n$, and a circuit F of size f, such that F distinguishes between $V^{S^*,T}_{\mathcal{C}(\kappa,n)}(i_1)$ and $V^{S^*,T}_{\mathcal{C}(\kappa,n)}(i_2)$ with an ϵ-advantage.

We say that the scheme $\mathcal{C}$ is (computationally) (S, T)*-private with privacy level* $(\mathcal{F}, \mathcal{E})$, if, for any function $\mathbf{f} \in \mathcal{F}$, there exists a function $\varepsilon \in \mathcal{E}$, such that for any κ, n the collusion (S, T) cannot $(\mathbf{f}(\kappa, n), \varepsilon(\kappa, n))$-break $\mathcal{C}(\kappa, n)$. In other words, every $\mathcal{F}$-bounded adversary corrupting (S, T) should gain from its view only an $\mathcal{E}$-bounded advantage in distinguishing between any two retrieval indices. Finally, we say that $\mathcal{C}$ is (s, t)*-private* (with privacy level $(\mathcal{F}, \mathcal{E})$), if it is (S, T)-private for all collusions (S, T) with $|S| = s$ and $|T| = t$. The parameters (s, t) will sometimes be referred to as the *privacy threshold* (in contrast to the *privacy level* $(\mathcal{F}, \mathcal{E})$). Since the default database privacy threshold considered in other PIR works is $t = 1$, in the context of commodity schemes "s-private" will stand for $(s, 1)$-private.

Note that $\mathcal{C}$ is information-theoretically (s, t) private if and only if it is computationally (s, t)-private with privacy level $(\mathcal{F}, \mathcal{E})$ for *all* function classes $\mathcal{F}, \mathcal{E}$. This observation will allow us to use the computational framework in theorems and proofs that apply to *both* the computational and the information-theoretic settings.

When referring to *specific* schemes the privacy level $(\mathcal{F}, \mathcal{E})$ will usually be omitted, under the implicit understanding that it is closely related to the strength of an underlying intractability assumption. As a default privacy level (which takes over whenever $\mathcal{F}, \mathcal{E}$ are omitted) $\mathcal{F}$ can be taken to be the class of all polynomials in κ (or equivalently in $\kappa + n$), and $\mathcal{E}$ to be the class of all functions $\varepsilon(\cdot, \cdot)$ which become negligible in κ whenever n is polynomially bounded in κ. That is, $\varepsilon \in \mathcal{E}$ if for any polynomial $p(\cdot)$ there is a negligible function $\varepsilon'(\cdot)$ such that $\varepsilon(\kappa, n) \leq \varepsilon'(\kappa)$ whenever $n \leq p(\kappa)$. This default definition corresponds to the usual "conservative" security assumptions which limit the

[8] It seems more natural to let the level of privacy depend on the security parameter κ alone. However, allowing the privacy level to depend on n as well better fits constructions (as in [9] and [7]) whose security slightly degrades with n, even when the adversary's resources are bounded by a fixed function of κ.

problem size and the adversary's power to be polynomial in the security parameter and the adversary's advantage to be negligible.

We finally remark that the above definition of computational privacy implies privacy in the sense of the single-parameter definition used in [9], where n serves both as a data size parameter and as a security parameter. Specifically, if a scheme $\mathcal{P}$ is private under our two-parameter definition with the default privacy level, then for any $c > 0$ the single-parameter scheme $\mathcal{P}_c$ defined by $\mathcal{P}_c(n) = \mathcal{P}(n^c, n)$ is private under the single-parameter definition. Moreover, if $\mathcal{P}$ is private with a stronger privacy level, then smaller functions of n can be substituted for κ, as small as *polylog*(n) in an extreme case (e.g., when $\mathcal{F} = \{2^{c_1\kappa}\}$ and $\mathcal{E} = \{2^{-c_2\kappa}\}$ for some constants $0 < c_1, c_2 < 1$).

2.3.3. *Complexity*

Complexity is measured, by default, in terms of communication. The *communication complexity* of a PIR scheme or a commodity scheme is denoted (α, β), where α (called the *query complexity*) is the maximal number of query bits sent from the user to any database, and β (called the *answer complexity*) is the maximal number of answer bits sent from any database to the user. The *reconstruction information complexity* of a PIR scheme $\mathcal{P}$, denoted γ, is the maximal length of the reconstruction information string z output by $\mathbf{que}_{\mathcal{P}}$.

Note that the communication complexity reflects only the communication cost of the retrieval stage. The *commodity complexity* of a commodity scheme is denoted (δ^u, δ^{db}), where δ^u (resp. δ^{db}) is the *maximal* number of commodity bits sent from any server to the user (resp. to any database). Since PIR schemes and commodity schemes are parameterized by the number of records n, a security parameter κ (in the computational case), and the record size ℓ (to be addressed in the next subsection), the complexity measures $\alpha, \beta, \gamma, \delta^u, \delta^{db}$ may depend on these parameters. Finally, whenever the parameter ℓ is omitted it is understood to be equal to 1. For instance, $\beta(\kappa, n)$ is used to denote the answer complexity on an *n-bit* data string with security parameter κ.

2.4. *Extending Bit Retrieval to Block Retrieval*

The definitions in Section 2.3 only address the default case of bit retrieval. A more general scheme $\mathcal{C}'$, allowing retrieval of multibit records (also referred to as *blocks*), may be defined by applying the following modifications to the original definitions. First, a record length parameter ℓ should be optionally given as an additional input to the algorithms comprising $\mathcal{C}'$ (as we shall see, this option is not used in our context). Second, the correctness definition should be strengthened to apply to every ℓ and $x \in (\{0, 1\}^\ell)^n$ (where $\mathbf{rec}_{\mathcal{C}'}$ should not be restricted to return a single bit). Finally, we require the privacy level to be independent of the record length ℓ; i.e., we use the same definitions except for extra universal quantifiers on ℓ where appropriate. In the remainder of this subsection we address the special case of block retrieval for PIR schemes; the more general case of commodity schemes can be handled similarly.

Let $\mathcal{P}$ be any PIR scheme, as defined in Section 2.3. We define a default extension of $\mathcal{P}$ into a block retrieval scheme $\mathcal{P}'$ in the following "naive" way, which is used in [10] as a basis for further optimization.[9] For any data string $x \in (\{0,1\}^\ell)^n$ and $1 \le w \le \ell$, let x^w denote the n-bit string obtained by taking only the wth bit from each record. To retrieve an ℓ-bit record using $\mathcal{P}'$: (1) the user invokes $\mathbf{que}_\mathcal{P}$ as in $\mathcal{P}$; (2) each database answers the user's query by invoking $\mathbf{ans}_\mathcal{P}$ ℓ times, once under each n-bit data string x^w; and (3) the user applies the reconstruction function $\mathbf{rec}_\mathcal{P}$ ℓ times, once for each answer.

Note that the user's queries in the scheme $\mathcal{P}'$ are independent of the record length ℓ. Hence, we have:

Claim 1 [10]. *Any PIR scheme $\mathcal{P}$ (for bit retrieval) can be extended into a private block retrieval scheme $\mathcal{P}'$, such that $\mathbf{que}_{\mathcal{P}'} = \mathbf{que}_\mathcal{P}$. Moreover, if the answer complexity of $\mathcal{P}$ is $\beta(\kappa, n)$, then the answer complexity of $\mathcal{P}'$ is $\beta'(\kappa, n, \ell) = \ell \cdot \beta(\kappa, n)$ (and the query complexity, reconstruction information complexity, and privacy level of $\mathcal{P}'$ are the same as of $\mathcal{P}$).*

Relying on Claim 1 we freely use any PIR scheme $\mathcal{P}$ (or similarly any commodity scheme $\mathcal{C}$) on data strings of arbitrary record size, and do not involve the record size in the privacy analysis.

3. PIR Schemes with Low Answer Complexity

Most of the commodity schemes constructed in this work can use any PIR scheme as a building block. However, for the commodity schemes to be efficient, we are typically interested in PIR schemes whose answer complexity is very low.

Table 1 summarizes the parameters of some PIR schemes whose answer complexity is minimized to either a single bit, in the multi-database case, or $\kappa^{O(1)}$ bits, in the computational single-database case. The parameters of some of these schemes will be explicitly referred to in what follows. The parameter d appearing in the table can be substituted by any positive integer (including 1). In the "security type" column, "i.t." stands for information-theoretic security, "comp." for computational security, "PRG" for the existence of a pseudo-random generator (or equivalently one-way functions [21]), "QRA" for the Quadratic Residuosity Assumption [20], "PRA" for the Prime Residuosity

Table 1. Parameters of some PIR schemes.

Name	k	t	α	β	γ	Security Type
$\mathcal{P}_1^k$	k	$k-1$	n	1	0	i.t.
$\mathcal{P}_2^{t,d}$	$td+1$	t	$O(n^{1/d})$	1	0	i.t.
$\mathcal{P}_3$	2	1	$\kappa \cdot 2^{O(\sqrt{\log n})}$	1	0	comp. (PRG)
$\mathcal{P}_4^d$	1	1	$O(d\kappa n^{1/d})$	κ^d	κ	comp. (QRA)
$\mathcal{P}_5^d$	1	1	$O(d\kappa n^{1/d})$	$\kappa \cdot 2^{O(d)}$	κ	comp. (PRA)
$\mathcal{P}_6$	1	1	$(\kappa + \log n)^{O(1)}$	$(\kappa + \log n)^{O(1)}$	$(\kappa + \log n)^{O(1)}$	comp. (Φ-H)

[9] We use schemes $\mathcal{P}$ with the smallest answer complexity possible; optimization techniques from [10] and [9] do not yield any improvement for such schemes.

Assumption (see [30] and references therein), and "Φ-H" stands for the newly introduced Φ-Hiding assumption (see [7]).

The scheme $\mathcal{P}_1^k$ is the simplest one to describe: The user picks k otherwise-random queries $q_1, \ldots, q_k \in \{0,1\}^n$ whose bitwise exclusive-or is equal to e_i, each database $\mathcal{DB}_j$ replies with the inner product (over GF(2)) $x \cdot q_j$, and the user reconstructs x_i by taking the exclusive-or of the k answer bits. (This is a simple generalization of an elementary scheme from [10].) The other schemes are variants of schemes from [10], [9], [23], [30], and [7].

$\mathcal{P}_2^{t,d}$ is obtained by applying a small optimization to the polynomial interpolation-based scheme from [10] (see Remark 2 in Section 6). $\mathcal{P}_3$ is a variant of the 2-database computational scheme from [9]; in this scheme the user's queries are interpreted as two short pseudorandom "seeds," which are expanded (independently) by the two databases to two n-bit strings whose exclusive-or is e_i. The scheme can then proceed as $\mathcal{P}_1^2$. Details of this scheme will appear in the journal version of [9].

The remaining schemes are all single-database schemes. $\mathcal{P}_4^d$ is a variant of the scheme from [23]. This variant and some optimized version of it (in a setting where a public random string is available) are described in Appendix 9.2. $\mathcal{P}_5^d$, which generalizes the construction of $\mathcal{P}_4^d$, is from [30]. Finally, $\mathcal{P}_6$ is from [7]. Since the main focus of this work is on obtaining general and provably secure reductions, we use the less efficient scheme $\mathcal{P}_4^d$ (which is based on a more "standard" security assumption) to instantiate our single-database results.

We stress that while the scheme $\mathcal{P}_6$ is essentially optimal as far as its asymptotic complexity is concerned, the relative performance of the different schemes under "real-life" parameters may vary. In particular, the information-theoretic schemes and the 2-database computational scheme $\mathcal{P}_3$ have a better communication complexity on small to moderately sized data strings (say, with $n = 10^6$), or on larger strings with larger records. Moreover, these schemes are significantly more computationally efficient than the single-database schemes, roughly corresponding to the efficiency difference between a private-key and a public-key encryption of the entire data.

4. Atomic Single-Server Commodity Schemes

In this section we present a simple transformation from any t-private k-database (computational or information-theoretic) PIR scheme to a $(0, t)$-private, single-server, k-database commodity scheme. Single-server schemes obtained via this transformation are referred to as *atomic schemes,* and are subsequently composed into schemes with improved privacy properties.

We start with an informal description of how atomic commodity schemes are constructed, where for simplicity we refer here to the single-database case; a formal treatment of the general case will follow.

Consider any 1-round single-database (computational) PIR scheme $\mathcal{P}$. Such a scheme may be viewed as the following three-stage procedure: (1) the user computes a randomized query q corresponding to the retrieval index i (to which we sometimes refer as *a query pointing to the* ith *data record*); (2) the database computes an answer to this query based on the database contents; and (3) the user reconstructs the ith data record, x_i, from the answer and some auxiliary reconstruction information z generated along with the query. While the communication and computation costs of each such step may vary from one scheme to another, none of the known PIR schemes is satisfactorily efficient in both of these aspects. The following simple idea allows us to shift most of the communication cost and a substantial part of the user's computation from the on-line protocol to an off-line stage, and from the user's hands to an external commodity server. Instead of having the user compute *on line* a query pointing to the desired data record, we let the server perform *off line* the following operations:

- Pick a random retrieval index r.
- Compute a random query q pointing to the rth data record, along with its associated reconstruction information z.
- Send the index r along with z to the user, and the query q to the database.

Such commodities supplied by the server can then serve as an *oblivious window*, pointing to a random location in the data string which is known to the user but is computationally hidden from the database. All that is left to the user, knowing the location of this window relative to his retrieval index, is to specify by how much the data string should be cyclically shifted (say, to the left) so that the desired record will be aligned with this window. Then, using the database's answer on the shifted data string and the reconstruction information supplied by the server, the user can efficiently reconstruct the desired data record. Note that since the privacy of $\mathcal{P}$ guarantees that r is kept private from the database, the shift amount $\Delta = i - r \pmod n$ sent by the user gives the database no useful information.

The procedure we have just described is referred to as the *atomic commodity scheme* based on $\mathcal{P}$, and is denoted $\mathcal{C}_{\mathcal{P}}$. Formalizing and generalizing the above procedure, we have:

Theorem 1. *Let $\mathcal{P}$ be any t-private, k-database PIR scheme ($k \geq 1$) with communication complexity (α, β) and reconstruction information complexity γ. Then there is a $(0, t)$-private, single-server, k-database commodity scheme $\mathcal{C}_{\mathcal{P}}$ with communication complexity $(\log n, \beta)$, commodity complexity $(\log n + \gamma, \alpha)$, and the same privacy level as $\mathcal{P}$.*[10]

Proof. A commodity scheme $\mathcal{C}_{\mathcal{P}}$ as required is formally described in Fig. 1. The correctness of $\mathcal{C}_{\mathcal{P}}$ follows from observing that when cyclically shifting x by Δ places to the left, the desired record x_i moves to position $i - \Delta = r$, to which the commodity queries point.

We turn to show that $\mathcal{C}_{\mathcal{P}}$ is $(0, t)$-private with the same privacy level as $\mathcal{P}$. Fix κ, n, and a collusion $T \subseteq [k]$ of t databases. We reduce the privacy of $\mathcal{C}_{\mathcal{P}}$ to the privacy of $\mathcal{P}$ by showing that if the collusion $(\emptyset, T)$ can (f, ϵ)-break $\mathcal{C}_{\mathcal{P}}(\kappa, n)$, then the collusion T can (f, ϵ)-break $\mathcal{P}(\kappa, n)$. Let $V_{\mathcal{C}}^{T}(i)$ denote the view of T-databases in $\mathcal{C}_{\mathcal{P}}(\kappa, n)$ on index i (more precisely, $V_{\mathcal{C}}^{T}(i)$ is the random variable $V_{\mathcal{C}_{\mathcal{P}}(\kappa,n)}^{S^*,T}(i)$, defined

[10] In particular, if $\mathcal{P}$ is information-theoretically private, then so is $\mathcal{C}_{\mathcal{P}}$.

```
Atomic-Scheme-$\mathcal{C}_\mathcal{P}$
        $\mathcal{P}$: a $k$-database PIR scheme
        $\mathcal{C}_\mathcal{P}$: a single-server $k$-database commodity scheme
$\mathbf{com}_{\mathcal{C}_\mathcal{P}}(1^\kappa, 1^n)$
    $r \overset{R}{\leftarrow} Z_n$;
    $((q_1, q_2, \ldots, q_k), z) \overset{R}{\leftarrow} \mathbf{que}_\mathcal{P}(1^\kappa, 1^n, r)$;
    return $((r, z), (q_1, \ldots, q_k))$;
$\mathbf{que}_{\mathcal{C}_\mathcal{P}}(1^n, i, (r, z))$
    $\Delta \leftarrow i - r \pmod n$;
    return $(\Delta, \Delta, \ldots, \Delta)$;
$\mathbf{ans}_{\mathcal{C}_\mathcal{P}}(j, x, \Delta, q_j)$
    return $a_j \overset{\text{def}}{=} \mathbf{ans}_\mathcal{P}(j, x \ll \Delta, q_j)$;
      (where $x \ll \Delta$ denotes a cyclic shift of $x$ by $\Delta$ records to the left).
$\mathbf{rec}_{\mathcal{C}_\mathcal{P}}((a_1, \ldots, a_k), (r, z))$
    return $\mathbf{rec}_\mathcal{P}((a_1, \ldots, a_k), z)$;
```

Fig. 1. Atomic single-server commodity scheme $\mathcal{C}_\mathcal{P}$.

in Section 2.3, with $S^* = \emptyset$), and similarly let $Q^T_\mathcal{P}(i)$ denote the view of T-databases in $\mathcal{P}$. Now, suppose that F is a circuit of size f distinguishing with an ϵ-advantage between $V^T_\mathcal{C}(i_1)$ and $V^T_\mathcal{C}(i_2)$, for some $i_1, i_2 \in Z_n$. By the definition of $\mathcal{C}_\mathcal{P}$, the view $V^T_\mathcal{C}$ (up to replicated components) is $V^T_\mathcal{C}(i) = (Q^T_\mathcal{P}(R), i - R)$, where R is a random variable uniformly distributed over Z_n. Since the random variable $(R, i - R)$ is distributed identically to $(i - R, R)$ (and since the randomness of $Q^T_\mathcal{P}$ is independent of R), the random variable $V^T_\mathcal{C}(i)$ is distributed identically to $(Q^T_\mathcal{P}(i - R), R)$. Now, since

$$|\mathbf{Pr}[F(Q^T_\mathcal{P}(i_1 - R), R) = 1] - \mathbf{Pr}[F(Q^T_\mathcal{P}(i_2 - R), R) = 1]| \geq \epsilon$$

then, using a standard averaging argument, there exists $r_0 \in Z_n$ such that this ϵ-advantage is maintained conditioned by $R = r_0$. That is,

$$|\mathbf{Pr}[F(Q^T_\mathcal{P}(i_1 - r_0), r_0) = 1] - \mathbf{Pr}[F(Q^T_\mathcal{P}(i_2 - r_0), r_0) = 1]| \geq \epsilon.$$

Therefore, the circuit F' defined by $F'(q) \overset{\text{def}}{=} F(q, r_0)$ is a circuit of size f distinguishing between retrieval indices $i_1 - r_0$ and $i_2 - r_0$ with an ϵ-advantage, as required.

Finally, the communication and commodity complexity of $\mathcal{C}_\mathcal{P}$ are clearly as specified, and if $\mathcal{P}$ is computationally efficient, then so is $\mathcal{C}_\mathcal{P}$. □

Note that total communication involving the user in $\mathcal{C}_\mathcal{P}$, counting both the off-line commodity distribution stage and the on-line retrieval stage, is dominated by the answer complexity of $\mathcal{P}$. Section 3 contains an overview of some known PIR schemes with low answer complexity. Such schemes, which are not very useful in the usual PIR setting, serve as the most natural building blocks for commodity schemes.

Finally, it is important to observe that in any atomic scheme $\mathcal{C}_\mathcal{P}$, a collusion of the server with a single database can easily learn i, regardless of the privacy threshold of $\mathcal{P}$. Moreover, even an honest server with a faulty source of randomness will compromise the user's privacy in $\mathcal{C}_{\mathcal{P}}$. This obvious weakness of atomic schemes is dealt with in the following two sections.

5. Composing Commodity Schemes

As observed above, in any atomic commodity scheme the user's privacy is totally dependent on a proper behavior of the single server. A natural approach for alleviating this problem is to distribute the user's trust among several servers rather than one. We achieve this by *composing* atomic commodity schemes into multiserver schemes with improved privacy properties.

5.1. *The Single-Database Case*

We start by describing the special case of composing *atomic, single-database* commodity schemes; a more general composition operator is defined and formally analyzed in the next subsection.

Consider two atomic single-database commodity schemes: $\mathcal{C}_{\mathcal{P}_1}$ with server $\mathcal{S}_1$, and $\mathcal{C}_{\mathcal{P}_2}$ with server $\mathcal{S}_2$. The composed scheme proceeds as follows:

COMMODITIES: Each of the two servers independently distributes commodities as in the corresponding atomic scheme. Let r_1, r_2 denote the random retrieval indices picked, respectively, by $\mathcal{S}_1, \mathcal{S}_2$, and let q_1, q_2 denote the corresponding $\mathcal{P}_1$- and $\mathcal{P}_2$-queries.

RETRIEVAL: The database simulates all possible queries made by the user in the retrieval stage of $\mathcal{C}_{\mathcal{P}_1}$, and constructs a virtual data string x' whose records consist of answers to these queries. Specifically, the lth record of x', $0 \leq l < n$, will consist of the answer according to $\mathcal{P}_1$ to the commodity-query q_1 on the original data string x shifted by l records to the left. The retrieval of the ith record of x can now be reduced to retrieval of the Δth record of x', where $\Delta = i - r_1 \pmod n$ is the query used in $\mathcal{C}_{\mathcal{P}_1}$ for retrieving the ith record of x. The user retrieves this Δth record of x' using the retrieval procedure of $\mathcal{C}_{\mathcal{P}_2}$, based on commodities supplied by $\mathcal{S}_2$. Knowing this record, the user can apply to it the reconstruction procedure of $\mathcal{C}_{\mathcal{P}_1}$ to obtain x_i. Note that x' has n records, exactly as the number of records in x. The larger record size of x' will only affect the database's answer, whose size may be proportional to this record size.[11]

The query sent by the user in the composed scheme is $i - r_1 - r_2$. Since both r_1 and r_2 are hidden from the database and exactly one of them is hidden from each server, i is kept private from any collusion of the database with a single server.

Intuitively, the transformation from the original data string x to the virtual data string x' corresponds to an *oblivious shift* of x by a random amount r_1, which is known to the user and $\mathcal{S}_1$ but is unknown to the database and $\mathcal{S}_2$. Indeed, each record of x' may be viewed as an encoding, according to $\mathcal{P}_1$, of a corresponding shifted record from x. Using this notion of oblivious shifts, the retrieval stage of the composed scheme described

[11] In the single-database case we consider the "naive" block retrieval of Claim 1 as a worst-case scenario. The complexity of the composed scheme can be substantially improved if $\mathcal{C}_2$ implements block retrieval in a more efficient way. Some amortization of the cost of retrieving blocks is obtained by the scheme from [30].

above can be viewed as follows:

- Using q_1, the database obliviously shifts x by r_1 records to obtain a virtual data string x'; then, using q_2, the database obliviously shifts x' by r_2 records to obtain a virtual data string x''.
- The user explicitly asks for the $(i - r_1 - r_2)$th record of x'', from which he can reconstruct x_i.

(Note that we have slightly modified the previous scheme; there the database only computes the single record of x'' required by the user.)

The above presentation makes it conceptually easy to generalize the two-server composed scheme into an m-server scheme, which keeps i private from any collusion of the database with $m - 1$ servers. In such an m-server scheme the database successively performs m oblivious shifts on the data, using commodities from the m different servers, and the user reconstructs x_i from the $(i - \Sigma_{h=1}^{m} r_h)$th record of the resultant virtual data string. Notice that with our default implementation of block retrieval (using Claim 1) each oblivious shift increases the record size of the virtual data string by a multiplicative factor equal to the answer size of the underlying PIR scheme. Thus, for all but very small values of m this approach will yield schemes with an unrealistically large answer complexity. One potential way of avoiding this problem is by using more efficient block retrieval techniques. This problem can also be avoided in the multi-database case, which is discussed in Sections 5.2 and 6.

Before introducing a more general multi-database composition operator, we state the result obtained by composing atomic single-database schemes in the manner described above.

Theorem 2. *Let $\mathcal{P}$ be a single-database PIR scheme with communication complexity (α, β) and reconstruction information complexity γ. Then, for any constant $m \geq 1$, there is an m-server, $(m - 1)$-private, single-database commodity scheme $\mathcal{C}_{\mathcal{P}}^{m}$, with communication complexity $(\log n, \beta^m)$, commodity complexity $(\log n + \gamma, \alpha)$, and the same privacy level as $\mathcal{P}$.*

A generalization of Theorem 2 is formally proved in the next subsection. As a special case, we may obtain the following:

Corollary 1. *For any constant integers $m, d \geq 1$ there is an m-server, single-database, $(m-1)$-private computational commodity scheme (assuming QRA), with communication complexity $(\log n, \kappa^{O(1)})$ and commodity complexity $(\log n, O(\kappa \cdot n^{1/d}))$.*

Proof. Such a scheme can be obtained by applying Theorem 2 to the PIR scheme $\mathcal{P}_4^d$. More precisely, the actual communication complexity is $(\log n, O(\kappa^{md}))$; for constant m and d, this is polynomial in κ.[12] □

[12] Fixing the number of databases [10], [1], or the complexity parameter d [23], has been the convention in other PIR related works. In Section 6 we present a (multi-database) scheme whose complexity is also polynomial in m.

We note that since the scheme $\mathcal{P}_4^d$ allows us to trade answer complexity for query complexity (an extreme case is $d = 1$, in which the query complexity is linear in n and the answer complexity is only κ), a similar tradeoff can be established between the *commodity* complexity and the *communication* complexity in the m-server scheme of Corollary 1. Finally, this scheme can be made more efficient if $\mathcal{P}_5^d$ or $\mathcal{P}_6$ is used instead of $\mathcal{P}_4^d$ (see Section 3).

5.2. *Multi-Database Composition*

Known multi-database PIR schemes possess several advantages over their single-database counterparts. First, they allow information-theoretic user privacy, which cannot be achieved at all in the single-database case (unless the entire database is sent to the user [10]). Other advantages are their computational complexity, which is typically much more modest, and their superior communication complexity on moderately sized data strings. Finally, and in our context most importantly, they can potentially have the smallest answer complexity possible—as low as a single bit (this is the case for the schemes $\mathcal{P}_1^k, \mathcal{P}_2^{t,d}, \mathcal{P}_3$). In contrast, it is not hard to observe that a very low answer complexity implies a poor level of computational privacy in single-database PIR schemes. Since the bottleneck of the previous multiserver solutions was the answer complexity of the underlying PIR schemes, multi-database schemes seem like better candidates for commodity schemes with a high threshold of server-privacy.

However, when trying to apply the composition technique described in the previous subsection to multi-database commodity schemes, the following problem arises. Consider an attempt to compose two atomic multi-database schemes, $\mathcal{C}_{\mathcal{P}_1}$ and $\mathcal{C}_{\mathcal{P}_2}$. When letting each database compute a virtual data string as defined for the single-database case, strings computed by different databases may differ; indeed, these strings depend on different $\mathcal{P}_1$-queries sent as commodities to the databases. Consequently, there is not enough data replication to allow using the multi-database scheme $\mathcal{P}_2$ for retrieval from the virtual data strings. The latter problem may be overcome by increasing the number of databases, thereby introducing sufficient additional data replication to allow the second-level retrieval. This idea is used in the following formalization of a composition operator, which generalizes the composition technique described in the previous subsection.

Consider any two commodity schemes, $\mathcal{C}_1$ and $\mathcal{C}_2$, where each $\mathcal{C}_b$ is an m_b-server k_b-database scheme with communication complexity (α_b, β_b) and commodity complexity $(\delta_b^u, \delta_b^{db})$. We define a composed scheme $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ using $m = m_1 + m_2$ servers and $k = k_1 k_2$ databases. For convenience, server indices will be taken from the set $(\{1\} \times [m_1]) \cup (\{2\} \times [m_2])$ and database indices from the set $[k_1] \times [k_2]$.

The composed scheme $\mathcal{C}$, on parameters $\pi \overset{\text{def}}{=} (\kappa, n)$, will invoke the scheme $\mathcal{C}_1$ on the same parameters π and the scheme $\mathcal{C}_2$ on the parameters $\pi' \overset{\text{def}}{=} (\kappa, n'(\pi))$, where $n'(\kappa, n)$ is the size of the query domain of $\mathcal{C}_1(\kappa, n)$. Note that if $\mathcal{C}_1$ is an atomic scheme, then $n'(\pi) = n$, and $n'(\pi) \approx 2^{\alpha_1(\pi)}$ in general. Hence, we require that $\alpha_1(\pi) = O(\log n)$ for $\mathcal{C}$ to be computationally efficient. The scheme $\mathcal{C}$ proceeds as follows.

COMMODITIES: Each server $\mathcal{S}_{1,h_1}$ generates commodities as $\mathcal{S}_{h_1}$ in $\mathcal{C}_1(\pi)$, except that each commodity originally sent from $\mathcal{S}_{h_1}$ to $\mathcal{DB}_{j_1}$ will now be sent from $\mathcal{S}_{1,h_1}$ to all databases $\mathcal{DB}_{j_1,j}$, $j \in [k_2]$. Similarly, each server $\mathcal{S}_{2,h_2}$ generates commodities as $\mathcal{S}_{h_2}$ in

$\mathcal{C}_2(\pi')$, except that each commodity originally sent from $\mathcal{S}_{h_2}$ to $\mathcal{DB}_{j_2}$ will now be sent from $\mathcal{S}_{2,h_2}$ to all $\mathcal{DB}_{j,j_2}$, $j \in [k_1]$.

RETRIEVAL:

1. The user computes k_1 queries $(q_1, \ldots, q_{k_1})$ pointing to the retrieval index i as in $\mathcal{C}_1(\pi)$ (with commodities from servers $\mathcal{S}_{1,h}$); then, viewing each query q_{j_1} as a retrieval index, the user computes k_2 queries $(q_{j_1,1}, \ldots, q_{j_1,k_2})$ pointing to q_{j_1} according to $\mathcal{C}_2(\pi')$ (using $\mathcal{C}_2$-commodities from the servers $\mathcal{S}_{2,h}$). Each query q_{j_1,j_2} is sent to the database $\mathcal{DB}_{j_1,j_2}$.
2. Each database $\mathcal{DB}_{j_1,j_2}$ computes a virtual data string $x^{(j_1)}$, consisting of n' records of size β_1, where each record contains an answer to a possible user's query in $\mathcal{C}_1$. Specifically, the lth record of $x^{(j_1)}$ is the answer, according to $\mathcal{C}_1$ (and using $\mathcal{C}_1$-commodities), to the lth retrieval query on x. The database $\mathcal{DB}_{j_1,j_2}$ replies to the user's query by simulating $\mathcal{C}_2$ on the data string $x^{(j_1)}$ and the user's query q_{j_1,j_2}.
3. The user reconstructs x_i by first recovering each entry $x^{(j_1)}_{q_{j_1}}$, $j_1 \in [k_1]$, from the answers of $\mathcal{DB}_{j_1,1}, \ldots, \mathcal{DB}_{j_1,k_2}$ (using the reconstruction function and commodities of $\mathcal{C}_2$), and then applying the reconstruction function of $\mathcal{C}_1$ to the resultant values.

A formal definition of the composed scheme $\mathcal{C}$ is given in Fig. 2. Its correctness follows directly from the correctness of $\mathcal{C}_1, \mathcal{C}_2$. The following lemma includes a straightforward complexity analysis.

```
**Composed-Scheme-$\mathcal{C}_1 \circ \mathcal{C}_2$**
        $\mathcal{C}_1$: an $m_1$-server $k_1$-database commodity scheme.
        $\mathcal{C}_2$: an $m_2$-server $k_2$-database commodity scheme.
        $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$: an $(m_1 + m_2)$-server $k_1 k_2$-database commodity scheme,
          with server indices $(\{1\} \times [m_1]) \cup (\{2\} \times [m_2])$ and database indices $[k_1] \times [k_2]$.
$\mathbf{com}_{\mathcal{C}}(1^\kappa, 1^n, (b, h))$    /* $b \in [2]$, $h \in [m_b]$ */
  $(c^u_{b,h}, (c^{db_1}_{b,h}, \ldots, c^{db_{k_b}}_{b,h})) \overset{R}{\leftarrow} \mathbf{com}_{\mathcal{C}_b}(1^\kappa, 1^{n_b}, h)$,
    where $n_1 = n$ and $n_2 = n'(\kappa, n)$ (the size of the query domain of $\mathcal{C}_1(\kappa, n)$).
  for all $(j_1, j_2) \in [k_1] \times [k_2]$,  $c^{db_{j_1,j_2}}_{b,h} \leftarrow c^{db_{j_b}}_{b,h}$;
  return $(c^u_{b,h}, (c^{db_{1,1}}_{b,h}, \ldots, c^{db_{k_1,k_2}}_{b,h}))$;
$\mathbf{que}_{\mathcal{C}}(1^\kappa, 1^n, i, (c^u_{1,1}, \ldots, c^u_{1,m_1}, c^u_{2,1}, \ldots, c^u_{2,m_2}))$
  $(q_1, \ldots, q_{k_1}) \leftarrow \mathbf{que}_{\mathcal{C}_1}(1^\kappa, 1^n, i, (c^u_{1,1}, \ldots, c^u_{1,m_1}))$;
  for all $j_1 \in [k_1]$,  $(q_{j_1,1}, \ldots, q_{j_1,k_2}) \leftarrow \mathbf{que}_{\mathcal{C}_2}(1^\kappa, 1^{n'(\kappa,n)}, q_{j_1}, (c^u_{2,1}, \ldots, c^u_{2,m_2}))$
    (where each string in the query domain of $\mathcal{C}_1(\kappa, n)$ is identified with an index in $Z_{n'}$);
  return $(q_{1,1}, \ldots, q_{k_1,k_2})$;
$\mathbf{ans}_{\mathcal{C}}((j_1, j_2), x, q_{j_1,j_2})$    /* $j_1 \in [k_1]$, $j_2 \in [k_2]$ */
  let $x^{(j_1)}$ be a virtual data string defined by:
    $x^{(j_1)}_{q_{j_1}} \leftarrow \mathbf{ans}_{\mathcal{C}_1}(j_1, x, q_{j_1}, (c^{db_{j_1,j_2}}_{1,1}, \ldots, c^{db_{j_1,j_2}}_{1,m_1}))$, $q_{j_1} \in Z_{n'}$;
  return $\mathbf{ans}_{\mathcal{C}_2}(j_2, x^{(j_1)}, q_{j_1,j_2}, (c^{db_{j_1,j_2}}_{2,1}, \ldots, c^{db_{j_1,j_2}}_{2,m_2}))$;
$\mathbf{rec}_{\mathcal{C}}((a_{1,1} \ldots, a_{k_1,k_2}), (c^u_{1,1}, \ldots, c^u_{1,m_1}, c^u_{2,1}, \ldots, c^u_{2,m_2}))$
  for all $j_1 \in [k_1]$,  $a_{j_1} \leftarrow \mathbf{rec}_{\mathcal{C}_2}((a_{j_1,1}, \ldots, a_{j_1,k_2}), (c^u_{2,1}, \ldots, c^u_{2,m_2}))$;
  return $\mathbf{rec}_{\mathcal{C}_1}((a_1, \ldots, a_{k_1}), (c^u_{1,1}, \ldots, c^u_{1,m_1}))$;
```

Fig. 2. Composed commodity scheme $\mathcal{C}_1 \circ \mathcal{C}_2$.

Lemma 1. *The communication complexity of $\mathcal{C}$ is $(\alpha_2(\pi'), \beta_1(\pi) \cdot \beta_2(\pi'))$, and its commodity complexity is* $(\max\{\delta_1^u(\pi), \delta_2^u(\pi')\}, \max\{\delta_1^{db}(\pi), \delta_2^{db}(\pi')\})$. *Furthermore, if the query domains of $\mathcal{C}_1, \mathcal{C}_2$ are of size n each, then $\pi = \pi' = (\kappa, n)$; hence in this case the communication complexity of $\mathcal{C}$ becomes* $(\log n, \beta_1 \cdot \beta_2)$ (*with query domain of size* n). *Finally, if $\alpha_1 = O(\log n)$, then $\mathcal{C}$ is computationally efficient.*

We turn to analyze the privacy of $\mathcal{C}$. Let $S = (\{1\} \times S_1) \cup (\{2\} \times S_2)$ be a set of corrupt servers, and let $T = T_1 \times T_2$ be a set of databases. We reduce the (S, T)-privacy of $\mathcal{C}$ to *both* the (S_1, T_1)-privacy of $\mathcal{C}_1$ and the (S_2, T_2)-privacy of $\mathcal{C}_2$. The reduction can be made tighter and cleaner in the case that $\mathcal{C}_1$ and $\mathcal{C}_2$ are either atomic schemes or compositions of atomic schemes. More generally, such a tighter reduction is possible whenever the composed schemes meet the following stronger privacy requirement.

Definition 1. We say that a *strong collusion* (S, T) can (f, ϵ)-break $\mathcal{C}(\kappa, n)$ if, for some circuit F of size f, indices $i_1, i_2 \in Z_n$, corruption strategy S^*, and *arbitrary* function *help*,

$$|\mathbf{Pr}[F(C_{\mathcal{C}}(i_1), help(Q_{\mathcal{C}}(i_1))) = 1] - \mathbf{Pr}[F(C_{\mathcal{C}}(i_2), help(Q_{\mathcal{C}}(i_2))) = 1]| \geq \epsilon,$$

where $C_{\mathcal{C}}(i) = C^{\bar{S},T}_{\mathcal{C}(\kappa,n)}(i)$ and $Q_{\mathcal{C}}(i) = Q^{S^*,T}_{\mathcal{C}(\kappa,n)}(i)$. We say that $\mathcal{C}$ is *strongly* (s, t)*-private* (with a specified privacy level) if it satisfies the privacy definition from Section 2.3 with respect to strong collusions.

Note that a strong collusion may perform an arbitrary (unbounded) computation *help* on the queries alone, followed by a bounded computation on the commodities and the output of *help*.

The following lemma may be proved very similarly to Theorem 1.

Lemma 2. *For any t-private PIR scheme $\mathcal{P}$, the atomic commodity scheme $\mathcal{C}_{\mathcal{P}}$ is* strongly $(0, t)$*-private with the same privacy level as $\mathcal{P}$.*

Lemma 3. *Fix κ, n* (*which determine π, n', π'*), *and suppose that the strong collusion* (*resp. collusion*) (S, T) *can (f, ϵ)-break $\mathcal{C}(\pi)$. Then*:

1. *The strong collusion* (*resp. collusion*) (S_1, T_1) *can (f, ϵ)-break $\mathcal{C}_1(\pi)$* (*resp.* $(f + f_2(\pi'), \epsilon)$*-break $\mathcal{C}_1(\pi)$, where $f_2(\pi')$ is the size of circuitry required for computing* $\mathbf{que}_{\mathcal{C}_2(\pi')}$).
2. *The strong collusion* (*resp. collusion*) (S_2, T_2) *can (f, ϵ)-break $\mathcal{C}_2(\pi')$.*

Proof. We use the following simplified notation. By $(C_1, C_2, Q_1(i, C_1), Q(i, C_1, C_2))$ we denote the joint random variables associated with the invocation of $\mathcal{C}(\pi)$, where C_1 and C_2 are, respectively, the $\mathcal{C}_1(\pi)$- and $\mathcal{C}_2(\pi')$-commodities, $Q_1(i, C_1)$ are the intermediate $\mathcal{C}_1$-queries computed by the user as a function of the index i and his commodities from C_1, and $Q(i, C_1, C_2)$ are the final $\mathcal{C}$-queries. We write $Q(i, C_1, C_2) = Q_2(Q_1(i, C_1), C_2)$, indicating that Q is obtained by applying $\mathbf{que}_{\mathcal{C}_2(\pi')}$, with commodities C_2, to each of the k_1 entries of Q_1. Finally, using the usual superscripts to denote restrictions of these variables or specify a corruption strategy, the view of the (S, T)-collusion with corruption strategy S^* is (up to replicated components)

$$((C_1^{\bar{S}_1, T_1}, C_2^{\bar{S}_2, T_2}), Q_2^{S_2^*, T_2}(Q_1^{S_1^*, T_1}(i, C_1), C_2)),$$

where S_b^* is the restriction of S^* to $\mathcal{C}_b$-servers.

If the strong collusion (S, T) can (f, ϵ)-break $\mathcal{C}(\pi)$, then there is a corruption strategy S^*, a circuit F of size f, indices $i_1, i_2 \in Z_n$, and function *help*, such that F distinguishes between

$$((C_1^{\bar{S}_1, T_1}, C_2^{\bar{S}_2, T_2}), \mathit{help}(Q_2^{S_2^*, T_2}(Q_1^{S_1^*, T_1}(i_b, C_1), C_2))), \tag{1}$$

$b = 1, 2$, with an ϵ-advantage. If an ordinary collusion (S, T) can (f, ϵ)-break $\mathcal{C}(\pi)$, then the above holds with *help* restricted to be the identity function.

We start by proving the first claim. Using an averaging argument, there exist some fixed commodities $\mathbf{c}_2$ output by all $\mathcal{C}_2$-servers (extending S_2^*) given which the ϵ-advantage of F is maintained. That is, conditioning by $\mathbf{c}_2$ and slightly bending corruption strategy notation, F distinguishes between

$$((C_1^{\bar{S}_1, T_1}, \mathbf{c}_2^{\bar{S}_2, T_2}), \mathit{help}(Q_2^{\mathbf{c}_2, T_2}(Q_1^{S_1^*, T_1}(i_b, C_1), \mathbf{c}_2))),$$

$b = 1, 2$, with an ϵ-advantage. Hence, letting $\mathit{help}'(q) = \mathit{help}(Q_2^{\mathbf{c}_2, T_2}(q, \mathbf{c}_2))$, there is a circuit of size f distinguishing between $(C_1^{\bar{S}_1, T_1}, \mathit{help}'(Q_1^{S_1^*, T_1}(i_b, C_1)))$, $b = 1, 2$, with an ϵ-advantage, implying that the strong collusion (S_1, T_1) can (f, ϵ)-break $\mathcal{C}_1$. The case of an ordinary collusion can be handled similarly: if *help* in (1) is the identity function, to $(f + f_2(\pi'), \epsilon)$-break $\mathcal{C}_1(\pi)$ one may use a circuit F_1 such that $F_1(\mathbf{c}, \mathbf{q}) = F((\mathbf{c}, \mathbf{c}_2^{\bar{S}_2, T_2}), Q_2^{\mathbf{c}_2, T_2}(\mathbf{q}, \mathbf{c}_2))$, where the evaluation of Q_2 can be handled with at most an $f_2(\pi')$ extra cost to the size of F.

Finally, to prove the second claim, we condition the view (1) by fixed $\mathcal{C}_1$-commodities $\mathbf{c}_1$ which maintain the ϵ-advantage of F. That is, the circuit F distinguishes between

$$((\mathbf{c}_1^{\bar{S}_1, T_1}, C_2^{\bar{S}_2, T_2}), \mathit{help}(Q_2^{S_2^*, T_2}(Q_1^{\mathbf{c}_1, T_1}(i_b, \mathbf{c}_1), C_2))),$$

$b = 1, 2$, with an ϵ-advantage. Letting $i'_b = Q_1^{\mathbf{c}_1, T_1}(i_b, \mathbf{c}_1)$, there is a circuit of size f distinguishing between $(C_2^{\bar{S}_2, T_2}, \mathit{help}(Q_2^{S_2^*, T_2}(i'_b, C_2)))$, $b = 1, 2$. We may conclude that the strong collusion (S_2, T_2) can (f, ϵ)-break $\mathcal{C}_2(\pi')$, and if *help* is the identity function, then this holds for an ordinary collusion as well. $\square$

Lemma 4. *If $\mathcal{C}_1$ is strongly (s_1, t)-private and $\mathcal{C}_2$ is strongly (s_2, t)-private, both with privacy level $(\mathcal{F}, \mathcal{E})$ and query domain of size n, then $\mathcal{C}$ is strongly $(s_1 + s_2 + 1, t)$-private with privacy level $(\mathcal{F}, \mathcal{E})$.*

Proof. Since both query domains are of size n, we have $\pi' = \pi = (\kappa, n)$. Now fix π, and suppose that some strong collusion (S, T), where $|S| = s_1 + s_2 + 1$ and $|T| = t$, can (f, ϵ)-break $\mathcal{C}(\pi)$. For $b = 1, 2$, let $S_b = \{h\colon (b, h) \in S\}$, and let $T_1 = \{j_1\colon \exists j\ (j_1, j) \in T\}$ and $T_2 = \{j_2\colon \exists j\ (j, j_2) \in T\}$. The strong collusion $(S, \tilde{T})$, where $\tilde{T} = T_1 \times T_2$, can also (f, ϵ)-break $\mathcal{C}(\pi)$, since $T \subseteq \tilde{T}$. Moreover, both $|T_1| \leq t$ and $|T_2| \leq t$, and either $|S_1| \leq s_1$ or $|S_2| \leq s_2$. It follows by Lemma 3 that in the first case ($|S_1| \leq s_1$) there is a strong (s_1, t)-collusion which can (f, ϵ)-break $\mathcal{C}_1(\pi)$, and in the second case ($|S_2| \leq s_2$) there is a strong (s_2, t)-collusion which can (f, ϵ)-break $\mathcal{C}_2(\pi)$. □

A direct application of the composition tool to atomic schemes of the previous section thus gives the following.

Theorem 3. *Let $\mathcal{P}^k$ be a t-private, k-database PIR scheme with communication complexity (α, β), and reconstruction information complexity γ. Then, for any constant $m \geq 1$, there is an m-server, $(m-1, t)$-private, k^m-database commodity scheme $\mathcal{C}^m_{\mathcal{P}^k}$, with communication complexity* $(\log n, \beta^m)$, *commodity complexity* $(\log n + \gamma, \alpha)$, *and the same privacy level as $\mathcal{P}^k$.*

Proof. A scheme $\mathcal{C}^m_{\mathcal{P}^k}$ as required can be obtained by composing m atomic commodity schemes based on $\mathcal{P}^k$ in an arbitrary order. Complexity, privacy, and computational efficiency (for a constant m) follow by induction from the claims about the composition operator. (Complexity and efficiency follow from Lemma 1 and privacy from Lemmas 2 and 3). For the sake of concreteness, an explicit description of such a composed scheme is given in Fig. 3. □

Remark 1. It can be readily verified that in the single-database case ($k = 1$), all servers in the composed scheme $\mathcal{C}^m_{\mathcal{P}^k}$ play a symmetric role (i.e., $\mathbf{com}_{\mathcal{C}}(\kappa, n, h)$ is independent

```
Composed-Scheme-$\mathcal{C}^m_{\mathcal{P}^k}$
        $\mathcal{P}^k$: a $k$-database PIR scheme
        $\mathcal{C} = \mathcal{C}^m_{\mathcal{P}^k}$: an $m$-server $k^m$-database commodity scheme with database names
          $\mathcal{DB}_\tau, \tau = \tau_1 \ldots \tau_m \in [k]^m$.
$\mathbf{com}_{\mathcal{C}}(1^\kappa, 1^n, h)$    /* $h \in [m]$ */
   $r_h \overset{R}{\leftarrow} Z_n$;
   $((q_h^1, \ldots, q_h^k), z_h) \overset{R}{\leftarrow} \mathbf{que}_{\mathcal{P}^k}(1^\kappa, 1^n, r_h)$;
   $c_h^u \leftarrow (r_h, z_h)$;
   for all $\tau \in [k]^m$, $c_h^{db_\tau} \leftarrow q_h^{\tau_h}$;
   return $(c_h^u, (c_h^{db_\tau})_{\tau \in [k]^m})$;
$\mathbf{que}_{\mathcal{C}}(1^n, i, ((r_1, z_1), \ldots, (r_m, z_m)))$
   $\Delta \leftarrow i - \sum_{h=1}^m r_h \pmod n$;
   return $(\Delta, \Delta, \ldots, \Delta)$;
$\mathbf{ans}_{\mathcal{C}}(\tau, x, \Delta, (q_1^{\tau_1}, \ldots, q_m^{\tau_m}))$    /* $\tau \in [k]^m$ */
   $x^\varepsilon \leftarrow x$;    /* $\varepsilon$ denotes the empty string */
   for $h \leftarrow 1$ to $m$, iteratively compute $n$-record data strings $x^{\tau'}$, such that $\tau' = \tau_1 \ldots \tau_h$,
      and $x_l^{\tau'} \leftarrow \mathbf{ans}_{\mathcal{P}^k}(\tau_h, x^{\tau_1 \cdots \tau_{h-1}} \ll l, q_h^{\tau_h})$, $l \in Z_n$;
   return $a_\tau \overset{\text{def}}{=} x_\Delta^\tau$;    /* only this entry of $x^\tau$ should be computed. */
$\mathbf{rec}_{\mathcal{C}}((a_1, \ldots, a_k), ((r_1, z_1), \ldots, (r_m, z_m)))$
   for $h \leftarrow m-1$ down-to 0, for all $\tau' \in [k]^h$, $a_{\tau'} \leftarrow \mathbf{rec}_{\mathcal{P}^k}((a_{\tau'1}, a_{\tau'2}, \ldots, a_{\tau'k}), z_{h+1})$;
   return $a_\varepsilon$;
```

Fig. 3. Composed m-server commodity scheme $\mathcal{C}^m_{\mathcal{P}^k}$.

of h). In the multi-database case, however, despite using the same commodity generation algorithm, each server sends its commodities to a different set of databases. By letting each server simulate all m servers, the servers' role can always be made symmetric at the expense of increasing the commodity complexity by a factor of m.

"Plugging in" the scheme $\mathcal{P}_3$ in Theorem 3, we obtain the following:

Corollary 2. *For any constant $m \geq 1$ there is an m-server, $(m-1)$-private, 2^m-database computational commodity scheme, with communication complexity $(\log n, 1)$ and commodity complexity $(\log n, \kappa \cdot 2^{O(\sqrt{\log n})})$ (assuming the existence of a pseudo-random generator).*

We remark that although the number of databases in the above corollary grows exponentially with the privacy threshold s, this overhead is arguably tolerable for small values of s such as 1 or 2.

6. Polynomial-Interpolation-Based Commodity Schemes

In all of the schemes obtained in the previous section, the total communication cost of retrieval grows exponentially with the server-privacy threshold s (though polynomial in $\log n$ and κ for a *fixed* s). This is clearly the case with the single-database scheme of Theorem 2, where the answer of this single database grows exponentially with m, but is also the case with schemes obtained via Theorem 3, where communication with each database may be only logarithmic in n (and independent of m when $\beta = 1$), but the number of databases grows exponentially with m.

In this section we extend techniques from [10], based on the method of low-degree polynomial interpolation (see [3] and [4]), to obtain multi-database commodity schemes which avoid this exponential growth of communication. In particular, achieving s-privacy would require $s+1$ servers, $s+2$ databases, and $\log n + 1$ communication with each database. This makes the total communication cost of retrieval grow only logarithmically in n and *linearly* in the privacy threshold s.

We use the following two lemmas.

Lemma 5. *Let n be an integer and q a prime power, let $\mathbf{y}^h$ represent a sequence of variables $y_0^h, y_1^h, \ldots, y_{n-1}^h$, and let δ_{i_1,i_2} denote Kronecker's function (i.e., δ_{i_1,i_2} equals 1 if $i_1 = i_2$ and 0 otherwise). Then for any $m \geq 1$ and $i \in Z_n$ there exists a degree-m multivariate polynomial $P_i^m(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^m)$ in $m \cdot n$ variables over $\mathrm{GF}(q)$, such that, for every $r_1, \ldots, r_m \in Z_n$,*

$$P_i^m(e_{r_1}, \ldots, e_{r_m}) = \delta_{i,r},$$

where $r = \sum_{h=1}^m r_h \pmod{n}$.

Moreover, P_i^m can be evaluated in polynomial time (in the size of its inputs).

Proof. Fixing n and q, define the following sequence of polynomials: $P_i^1(\mathbf{y}^1) = y_i^1$ and

$$P_i^m(\mathbf{y}^1, \ldots, \mathbf{y}^m) = \sum_{w \in Z_n} P_w^{m-1}(\mathbf{y}^1, \ldots, \mathbf{y}^{m-1}) \cdot y_{i-w}^m,$$

where the subtraction $i - w$ is taken modulo n. It easily follows by induction on h that P_i^m as defined above meets the specified requirements. Since $\mathbf{P}^h \overset{\text{def}}{=} (P_0^h, \ldots, P_{n-1}^h)$ can be efficiently evaluated given the values of $\mathbf{P}^{h-1}$, the values of $\mathbf{P}^m$ can be efficiently computed on a given assignment by iterating the evaluation of all $\mathbf{P}^h$, where h runs from 1 to m. □

The next lemma slightly improves a similar bound implicit in [10].

Lemma 6. *Let l, d be positive integers, and let $q > d+1$ be a prime power. Then there exist $n_{l,d} \overset{\text{def}}{=} \binom{l+d}{d}$ degree-d multivariate polynomials $p^i(y_1, \ldots, y_l)$, $0 \le i < n_{l,d}$, and assignments $\mathbf{v}^i \in \mathrm{GF}(q)^l$, $0 \le i < n_{l,d}$, such that $p^{i_1}(\mathbf{v}^{i_2}) = \delta_{i_1,i_2}$ for all $0 \le i_1, i_2 < n_{l,d}$.*

Proof. The existence of such $p^i, \mathbf{v}^i$ can be easily derived from the following facts:

- the number of degree-d *monic monomials*[13] over $y_1, \ldots, y_l$ is $\binom{l+d}{d}$ (as the number of ways for placing at most d identical balls in l distinct bins);
- when $d < q - 1$ these monomials are linearly independent, where each monomial p is identified in a natural way with the vector $\mathbf{u}^p \in \mathrm{GF}(q)^{q^l}$ such that $u^p_{y_1 \cdots y_l} = p(y_1, \ldots, y_l)$.

Now, since the $n_{l,d} \times q^l$ matrix whose rows are all the vectors $\mathbf{u}^p$ is of full rank,[14] it is row-equivalent to a matrix A of which $n_{l,d}$ columns induce an identity matrix. Identifying each of these $n_{l,d}$ columns with an assignment $\mathbf{v}^i$ and each linear combination used for obtaining a row of A with a corresponding polynomial p^i, the desired result is obtained.

An explicit construction of such $p^i, \mathbf{v}^i$, slightly improving a construction from [10],[15] is described in the following. Let $m^i(y_1, \ldots, y_l)$ be the ith degree-d monic monomial (say, according to lexicographic order). With each m^i associate a "characteristic vector" $\mathbf{v}^i = (v_1^i, \ldots, v_l^i)$, such that $m^i = \prod_{j=1}^l y_j^{v_j^i}$. Letting $y_0 \overset{\text{def}}{=} d - \sum_{j=1}^l y_j$ and $v_0^i \overset{\text{def}}{=} d - \sum_{j=1}^l v_j^i$ $(= d - \deg(m^i))$, define p^i as

$$p^i(y_1, \ldots, y_l) = \prod_{j=0}^{l} \prod_{k=0}^{v_j^i - 1} \frac{y_j - k}{v_j^i - k}.$$

[13] We define a degree-d monic monomial to be the product of *at most* d, not necessarily distinct, variables; "monic" indicates that the coefficient is 1.

[14] Here and in the following, an $a \times b$ matrix is said to be of full rank if its rank is equal to $\min(a, b)$.

[15] The construction in [10] implies a similar bound with $n_{l,d} = \binom{l+d-1}{d}$, utilizing only the $\binom{l+d-1}{d}$ monomials whose degree is *exactly* d.

Since $\sum_{j=0}^{l} v_j^i = d$, each p^i is of degree d. It is straightforward to verify that the constructed p^i, $\mathbf{v}^i$ meet the requirements. $\square$

It is interesting to note that the bound $\binom{l+d}{d}$ in the lemma is tight, as it coincides with the dimension of the linear space of degree-d multivariate polynomials (which is spanned by the degree-d monic monomials). This means that the application of the polynomial interpolation technique to PIR, as in [10], in a sense cannot be pushed any further.

Theorem 4. *Let m, t, d be positive integers, let $k \stackrel{\text{def}}{=} mtd + 1$ and q be a prime power greater than $k + 1$. Let $l_{n,d}$ denote the smallest integer l such that $\binom{l+d}{d} \geq n$. Then there is an $(m - 1, t)$-private information-theoretic commodity scheme $\mathcal{C}_{m,t,d}$, with m servers, k databases, communication complexity* $(\log n, \log q)$, *and commodity complexity* $(\log n, l_{n,d} \cdot \log q)$. *Moreover, this scheme can be applied to data strings whose records are elements of* $\mathrm{GF}(q)$ (*rather than single bits*) *at the same cost.*

Proof. Let $k = mtd + 1$ and $l = l_{n,d}$, let p^i, $\mathbf{v}^i$ be as promised by Lemma 6, and let $\mathbf{P}^m$ be as promised by Lemma 5. We view the data bits (or records) as elements of $\mathrm{GF}(q)$. A commodity scheme $\mathcal{C}_{m,t,d}$ as required is described in the following:

COMMODITIES: Each server $\mathcal{S}_h$, $1 \leq h \leq m$:

1. Picks a random index $r_h \in Z_n$, which is sent as a commodity to the user, and computes the corresponding assignment $\mathbf{v}^{r_h}$.
2. Independently shares each entry of $\mathbf{v}^{r_h}$ according to Shamir's secret sharing scheme [29] with privacy threshold t, over $\mathrm{GF}(q)$. Formally, for each wth entry $v_w^{r_h}$, $1 \leq w \leq l$, and each database $\mathcal{DB}_j$, $\mathcal{S}_h$ sends to $\mathcal{DB}_j$ the share $f_w^h(\theta_j)$, where f_w^h is a random degree-t (univariate) polynomial with free coefficient $v_w^{r_h}$, and each θ_j, $1 \leq j \leq k$, is a distinct nonzero element in $\mathrm{GF}(q)$ associated with $\mathcal{DB}_j$. We let $\mu^{h,j}$ denote the l-tuple of shares sent from $\mathcal{S}_h$ to $\mathcal{DB}_j$.

RETRIEVAL:

1. $\mathcal{U}$ sends to each database the query $\Delta \stackrel{\text{def}}{=} i - \sum_{h=1}^{m} r_h \pmod{n}$.
2. Each database $\mathcal{DB}_j$ replies with
$$a_j \stackrel{\text{def}}{=} \sum_{w \in Z_n} x_{w+\Delta} \cdot P_w^m(\mathbf{p}(\mu^{1,j}), \ldots, \mathbf{p}(\mu^{m,j})),$$
where $\mathbf{p} = (p^0, p^1, \ldots, p^{n-1})$, and $w + \Delta$ is computed modulo n.
3. $\mathcal{U}$ reconstructs by interpolation: x_i is taken to be the free coefficient of the (unique) degree-mtd univariate polynomial p over $\mathrm{GF}(q)$ such that $p(\theta_j) = a_j$, $j = 1, \ldots, k$.

PRIVACY: Let $S = [m] \setminus \{h_0\}$ be a set of $m - 1$ corrupt servers with corruption strategy S^*, and let $T \subseteq [k]$ be a set of t corrupt databases. The privacy of the scheme follows from the fact that the collusion S, T cannot obtain any information about the index r_{h_0} picked by the remaining server. More formally, for any n, i the view $V^{S^*,T}_{\mathcal{C}_{m,t,d}}(n,i)$ includes:

- commodities $c^T_{h_0}$ sent by the incorrupt server $\mathcal{S}_{h_0}$ to databases from T;
- the user's query $\Delta = i - \sum_{h=1}^{m} r_h \pmod{n}$, where all indices r_h except r_{h_0} are determined by corrupt servers (as specified by S^*).

Now, the commodities $c^T_{h_0}$ consist of l independent t-tuples of elements from $\mathrm{GF}(q)$, each containing t shares generated by a t-private Shamir's secret-sharing. It follows that $c^T_{h_0}$ is distributed uniformly over $\mathrm{GF}(q)^{lt}$, independently of r_{h_0}. Since r_{h_0} is uniformly distributed over Z_n, we may conclude that the joint view $(c^T_{h_0}, \Delta)$ is uniformly distributed over $\mathrm{GF}(q)^{lt} \times Z_n$, independently of i.

CORRECTNESS: It suffices to show that the points (θ_j, a_j), $j \in [k]$, lie on a degree-mtd (univariate) polynomial whose free coefficient is x_i. This can be argued in a straightforward way by tracing the computation of the answers a_j. For each $h \in [m]$ and $u \in [l]$, the points $(\theta_j, \mu^{h,j}_u)$, $j \in [k]$, lie on a degree-t polynomial (namely, the polynomial f^h_u picked by the user) whose free coefficient is $v^{r_h}_u$. Since each p^w is of degree d, for any $h \in [m]$ and $w \in Z_n$ the points $(\theta_j, p^w(\mu^{h,j}))$ lie on a degree-td polynomial whose free coefficient is $p^w(\mathbf{v}^{r_h})$, which by Lemma 6 equals δ_{e,r_h}. Finally, since each P^m_w is of degree d, for each $w \in Z_n$ the points $(\theta_j, P^m_w(\mathbf{p}(\mu^{1,j}), \ldots, \mathbf{p}(\mu^{m,j})))$ lie on a degree-mtd polynomial whose free coefficient is $P^m_w(e_{r_1}, \ldots, e_{r_m})$, which by Lemma 5 is equal to $\delta_{w,r}$ (where $r = \sum r_h$). It follows that the points (θ_j, a_j) lie on a degree-mtd polynomial whose free coefficient is $\sum_{w \in Z_n} x_{w+\Delta} \cdot \delta_{w,r} = x_{r+\Delta} = x_i$. This concludes the proof of Theorem 4. □

Remark 2. When retrieving a single-bit, the scheme $\mathcal{C}_{m,t,d}$ can be converted into a similar scheme, in which each database replies with a *single* answer bit, and the user takes the exclusive-or of the answers to obtain x_i. We briefly describe how this is done. Observe that in $\mathcal{C}_{m,t,d}$ the user reconstructs x_i by computing a *fixed* linear combination over $\mathrm{GF}(q)$ of the k field elements replied by the databases. Thus, as a first step we can let each database multiply its original answer by the corresponding coefficient, so that reconstruction consists of computing the *sum* of all answers over $\mathrm{GF}(q)$. Then, if q is chosen to be a power of 2 ($q = 2^{\lceil \log(k+1) \rceil}$ will suffice) it is enough to send the user only the "least significant bit" of each answer.

Combining the above remark with the fact that $l_{n,d} = O(n^{1/d})$ for any constant d, we have the following corollary of Theorem 4:

Corollary 3. *For any constants s, t, d there is an (s,t)-private information-theoretic commodity scheme with $s+1$ servers, $k = dt(s+1)+1$ databases, communication complexity* $(\log n, 1)$ *and commodity complexity* $(\log n, O(n^{1/d}))$.

7. Multiple-Query Schemes

In this section we show that the commodity complexity of previous schemes can be amortized over multiple queries made by the user.

A ρ-*query* PIR or commodity scheme can be defined using a straightforward extension of the single-query definitions; the generalized privacy requirement should assert that any two retrieval *index vectors* $\mathbf{i} = (i_1, \ldots, i_\rho)$ and $\mathbf{i}' = (i'_1, \ldots, i'_\rho)$ cannot be distinguished (in the appropriate sense) by the adversary. Any commodity scheme $\mathcal{C}$ (or PIR scheme $\mathcal{P}$) can be extended into a ρ-query scheme using ρ parallel and independent repetitions. This extension is referred to as the *naive q-query extension* of $\mathcal{C}$. It follows by a standard hybrid argument (see [14]) that if $\mathcal{C}$ is private with privacy level $(\mathcal{F}, \mathcal{E})$, then its naive ρ-query extension is private with privacy level $(\mathcal{F}, \rho\mathcal{E})$, where $\rho\mathcal{E} \stackrel{\text{def}}{=} \{\rho\varepsilon \colon \varepsilon \in \mathcal{E}\}$. We show that in the case of our commodity schemes, the commodity cost of the naive ρ-query extension can be reduced.

We start with a motivating example. Suppose that the user wishes to retrieve two records, with (arbitrary) indices i_1, i_2, using a 1-private single-database commodity scheme $\mathcal{C}^2_{\mathcal{P}}$, where $\mathcal{P}$ is some single-database PIR scheme. In the naive 2-query extension of $\mathcal{C}^2_{\mathcal{P}}$, the scheme is independently invoked twice in parallel. The retrieval cost of this solution is twice as large as that for a single query, and so is its commodity cost. The total number of commodity pairs c^u, c^{db} generated by the two servers will thus be four (each server generates two pairs, one for each retrieval). Note that one cannot use the same commodities for the two retrievals, since this would reveal the difference $i_1 - i_2$ to the database, potentially disclosing too much information about what the user is looking for. We now show that using an additional server, the *total* commodity cost of the above scheme can be improved to three commodity pairs of the same size as before. Consider a scheme in which each of the three servers $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ sends a single commodity pair, as in the original single-query scheme, and then i_1 is retrieved using the scheme $\mathcal{C}^2_{\mathcal{P}}$ with commodities from $\mathcal{S}_1, \mathcal{S}_2$, and i_2 is retrieved using the same scheme with commodities from $\mathcal{S}_2, \mathcal{S}_3$. The view of the database will consist of the three commodities supplied by the different servers, as well as the user's queries $i_1 - r_1 - r_2$ and $i_2 - r_2 - r_3$. It is not hard to verify that the joint distribution of these queries reveals nothing about (i_1, i_2) as long as at least two of r_1, r_2, r_3 are kept private. Assuming that at most one server is dishonest, the (computational) privacy of at least two of the three indices is ensured. Summarizing, we have obtained a 1-private 3-server scheme for retrieving two records, with the same retrieval complexity as the naive 2-server scheme, but with a lower commodity cost (three commodities instead of four). In the following we show how this can be generalized to obtain substantial savings in the commodity cost, asymptotically by up to a multiplicative factor of $s + 1$.

Theorem 5. *Assume n is a prime power, and let G be a full-rank $\rho \times m$ matrix over* $\mathrm{GF}(n)$ *such that the Hamming weight of every row in G is w, and G generates a linear code whose minimal distance is d. Let $\mathcal{C}^w$ be a commodity scheme obtained via Theorems* 2 *or* 4; *in particular, $\mathcal{C}^w$ is a w-server, k-database,* $(w - 1, t)$*-private commodity scheme with communication complexity* $(\log n, \beta)$ *and commodity complexity* $(\log n, \delta^{db})$, *in which all servers play a symmetric role. Then there is an m-server, k-database,* $(d - 1, t)$*-private commodity scheme $\mathcal{C}^m_\rho$ for ρ retrievals, with communication complexity* $(\rho \log n, \rho\beta)$ *and commodity complexity* $(\log n, \delta^{db})$. *Moreover, $\mathcal{C}^m_\rho$ has the privacy level of the naive ρ-query extension of its underlying PIR scheme $\mathcal{P}$* (*and is information-theoretically private if $\mathcal{C}^w$ is obtained via Theorem* 4).

Proof. Observe that in every scheme $\mathcal{C}^w$ as above, the user's query is of the form $i-\sum_{h=1}^{w} r_h$, where each r_h is supplied by a different server. We denote by $\mathcal{C}^w[\lambda_1, \dots, \lambda_w]$, where $\lambda_1, \dots, \lambda_w$ are fixed nonzero elements of GF(n), a generalization of $\mathcal{C}^w$ in which the user's query is $i - \sum_{h=1}^{w} \lambda_h r_h$ (i.e., $\mathcal{C}^w = \mathcal{C}^w[1, 1, \dots, 1]$). In case of a scheme $\mathcal{C}^w$ obtained via composition of w atomic single-database schemes (Theorem 2), such a generalization can be realized by modifying the definition of the hth composed atomic scheme so that the user's query is $\Delta = i - \lambda_h r$ (instead of $i - r$), and each database replies with an answer on a database x' such that $x'_j = x_{\lambda_h \cdot j+\Delta}$ (instead of replying on x cyclically shifted by Δ). Schemes obtained via the polynomial interpolation technique (Theorem 4) can be appropriately generalized by a straightforward modification of the polynomials P_i^m from Lemma 5.

We now define the scheme $\mathcal{C}_\rho^m$.

COMMODITIES: Each server $\mathcal{S}_h$, $1 \le h \le m$, sends commodities as a single server in $\mathcal{C}^w$.

RETRIEVAL: Let $g_1^u, \dots, g_w^u$ denote the nonzero entries in the uth row of G, and $h_1^u, \dots, h_w^u$ their corresponding columns. Then, for each retrieval index i_u, $1 \le u \le \rho$, the user and the databases execute the retrieval protocol of $\mathcal{C}^w[g_1^u, \dots, g_w^u]$, using commodities supplied by $\mathcal{S}_{h_1^u}, \dots, \mathcal{S}_{h_w^u}$.

The correctness of $\mathcal{C}_\rho^m$ follows directly from the correctness of the schemes $\mathcal{C}^w$ and from the fact that all servers in $\mathcal{C}^w$ play a symmetric role. Since each of the m servers sends commodities for a single retrieval, as in $\mathcal{C}^w$, the commodity complexity is as indicated. The communication complexity is the same as that of ρ retrievals using $\mathcal{C}^w$.

It remains to show that the scheme $\mathcal{C}_\rho^m$ is $(d-1, t)$-private. Let S be a set of $d-1$ corrupt servers and let T be a set of t databases. Since G generates a linear code with minimal distance d, the $\rho \times (m-d+1)$ matrix $G_{\bar{S}}$, obtained by restricting G to its columns with indices from $\bar{S}$, is of full rank (otherwise there exists a nonzero codeword whose Hamming weight is smaller than d). It follows that there is a server set S' of size $m-\rho$, $S \subseteq S'$, such that the (square) matrix $G_{\bar{S}'}$ is nonsingular. If the collusion (S, T) can (f, ϵ)-break $\mathcal{C}_\rho^m(\kappa, n)$, then the same holds for the (larger) collusion (S', T).

Now, fix κ, n, and corruption strategy S'^*, and let $\mathbf{R} = (R_1, \dots, R_m)$ be a random variable consisting of the indices sent as commodities to the user. Note that the entries R_h with $h \in \bar{S}'$ are uniformly and independently distributed over GF(n), and each remaining entry R_h, $h \in S'$, has some fixed value r_h determined by $S'^*(\kappa, n)$. The user's query (to each database) is $\mathbf{i} - G\mathbf{R}$, where $\mathbf{i} = (i_1, \dots, i_\rho)$ is his index vector. The commodities sent from servers in $\bar{S}'$ to databases in T include queries from $|\bar{S}'|$ independent invocations of an underlying PIR scheme $\mathcal{P}(\kappa, n)$, where each invocation uses a corresponding entry of $\mathbf{R}$ as its retrieval index; we denote this joint distribution of commodities by $Q_{\mathcal{P}}(R_{\bar{S}'})$. The joint view of databases from T on index vector $\mathbf{i}$ is $V_{\mathcal{C}}(\mathbf{i}) = (Q_{\mathcal{P}}(\mathbf{R}), \mathbf{i} - G\mathbf{R})$.

Now, suppose there are two index vectors $\mathbf{i}_1, \mathbf{i}_2$ and a circuit F of size f such that F distinguishes between $V_{\mathcal{C}}(\mathbf{i}_1)$ and $V_{\mathcal{C}}(\mathbf{i}_2)$ with an ϵ-advantage. That is, F distinguishes with an ϵ-advantage between $(Q_{\mathcal{P}}(\mathbf{R}_{\bar{S}'})Q_{\mathcal{P}}(\mathbf{r}_{S'}), \mathbf{i}_b - G_{\bar{S}'}\mathbf{R}_{\bar{S}'} - G_{S'}\mathbf{r}_{S'})$, $b = 1, 2$. Using the independence of $Q_{\mathcal{P}}(\mathbf{R}_{\bar{S}'})$ and $Q_{\mathcal{P}}(\mathbf{r}_{S'})$, there exists a circuit F' of size f distinguishing with an ϵ-advantage between $(Q_{\mathcal{P}}(\mathbf{R}_{\bar{S}'}), \mathbf{i}'_b - G_{\bar{S}'}\mathbf{R}_{\bar{S}'})$, $b = 1, 2$, where $\mathbf{i}'_b = \mathbf{i}_b - G_{S'}\mathbf{r}_{S'}$. Finally, since for any index vector $\mathbf{i}$ the random variable $(Q_{\mathcal{P}}(\mathbf{R}_{\bar{S}'}), \mathbf{i} - G_{\bar{S}'}\mathbf{R}_{\bar{S}'})$ is identically distributed to $(Q_{\mathcal{P}}(G_{\bar{S}'}^{-1}(\mathbf{i} - \mathbf{R}_{\bar{S}'})), \mathbf{R}_{\bar{S}'})$, we may apply yet another averaging argument to conclude that for some index vectors $\mathbf{i}''_1, \mathbf{i}''_2$ there is a circuit F'' of size f

distinguishing between $Q_{\mathcal{P}}(\mathbf{i}''_1)$ and $Q_{\mathcal{P}}(\mathbf{i}''_2)$ with an ϵ-advantage. Hence we have shown that $\mathcal{C}^m_\rho$ is $(d-1, t)$-private, with the same privacy level as that of the naive ρ-query extension of its underlying PIR scheme $\mathcal{P}$. □

Remark 3. The condition on G in Theorem 5 can be relaxed to allow rows with different Hamming weight in G; in such cases, w can be taken as the *maximal* row weight in G. This generalization, however, is not very useful for our purposes.

In the following we focus on the case where $w = d$, in which Theorem 5 induces no penalty in the communication cost or the number of databases of the multiquery scheme. This restriction motivates the following problem: Given a prime power n and positive integers ρ, d, find a minimal-length linear code over GF(n) which is generated by ρ linearly independent codewords of weight d and whose minimal distance is d. We let $m(n, \rho, d)$ denote this minimal length, corresponding to the minimal number of commodity-tuples which by Theorem 5 are sufficient for performing ρ independent $(d-1)$-private retrievals from an n-record data string, with no penalty in the communication complexity or the number of databases. The commodity cost $m(n, \rho, d)$ should be compared with the cost of the corresponding naive extension scheme, whose d servers distribute a total of $d\rho$ commodity-tuples. For instance, $m(n, \rho, 2) = \rho + 1$, as the $\rho \times (\rho + 1)$ matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ & & & \ddots & & \\ 0 & 0 & \ldots & 0 & 1 & 1 \end{pmatrix}$$

generates (over GF(n)) a code of distance 2, thus generalizing the motivating example from the beginning of the section to an asymptotic savings factor of 2 for 1-private schemes.

More generally, we have:

Fact 1. *For any n, ρ, d such that $n \geq \rho - 1$, $m(n, \rho, d) = \rho + d - 1$.*

Proof. For n, ρ, d as above, there exist $[\rho + d - 1, \rho, d]$ linear codes over GF(n) (see Chapter 11 of [25]). The $\rho \times (\rho + d - 1)$ generating matrix G of such code can be transformed via elementary row operations to a matrix G' generating the same code, which contains a $\rho \times \rho$ identity submatrix. Since the Hamming weight of each row of G' is at most $(\rho + d - 1) - \rho + 1 = d$, we have shown that $m(n, \rho, d) \geq \rho + d - 1$. On the other hand, it follows from the Singleton bound (see p. 33 of [25]) that $m(n, \rho, d) \leq \rho + d - 1$. □

We remark that the requirement $n \geq \rho - 1$ is necessary for the above bound to hold. Luckily, in most plausible situations n is significantly larger than ρ,[16] in which case the following corollary of Fact 1 applies.

[16] Frequently changing *small* databases can yield exceptions to this rule.

Corollary 4. *Assuming that the number of queries ρ is smaller than the database size n, Theorem* 5 *can asymptotically save a factor of $s+1$ in the amortized commodity cost of multiquery (s,t)-private schemes obtained via Theorems* 2, 3, *or* 4. *If the number of servers is limited to m_0, $m_0 > s$, the amortized savings factor is $(s+1)\cdot((m_0-s)/m_0)$.*

8. Commodity Testing

So far we have only addressed the goal of protecting the user's privacy, without considering issues of correctness in the presence of faulty parties. In this section we consider the problem of *commodity testing*, that is, verifying whether commodities provided by a given server are valid.

We restrict our attention to commodities for which there exists a PIR scheme $\mathcal{P}$, such that the user's commodity is some retrieval index r (possibly along with reconstruction information), and the databases' commodities consist of queries, generated according to $\mathcal{P}$, pointing to r. We note that commodities used in atomic schemes, and hence also in the composition of such schemes, are of this type. *Correctness* of such commodities is defined as follows.

Definition 2. Given a PIR scheme $\mathcal{P}$, data size n, and corresponding commodities $c = ((r,z),(q_1,\ldots,q_k))$ (supposedly output by $\mathbf{com}_{\mathcal{C}_{\mathcal{P}}(\kappa,n)}$ for some κ), the commodities c are said to be *correct on a data string* x, $x \in \{0,1\}^n$, if $\mathbf{rec}_{\mathcal{P}}((a_1,\ldots,a_k),z) = x_r$, where $a_j = \mathbf{ans}_{\mathcal{P}}(j,x,q_j)$; their *correctness ratio* is the proportion of data strings on which they are correct. The commodities are said to be *correct* if they are correct on all data strings of length n.

Notice that in any commodity scheme which is composed of atomic schemes, ensuring correctness of all commodities distributed by the servers guarantees correct execution of the retrieval procedure, assuming that the databases are honest.

We give two types of procedures for testing correctness of commodities, the second being more general than the first; however, procedures of the first type are much more efficient, and despite their lack of generality can be applied to most PIR schemes known to date. Both procedures treat the underlying PIR scheme as a black box, verifying correctness of commodities by testing them on some (small) sample of data strings. While their validity relies on the honest behavior of the databases, none of them compromises the user's privacy, even when there are dishonest databases. Finally, both procedures require a single round of (off-line) interaction, and their communication complexity involves an error probability parameter ϵ.

In the remainder of this section we describe the more efficient (and less general) testing procedure. The more general type is discussed in Appendix B.

8.1. *Linear Schemes*

Fix a security parameter κ, data size n, and a finite field $\mathbf{F}$. A PIR scheme $\mathcal{P}(\kappa,n)$ is said to be *linear over* $\mathbf{F}$ if for any strings $q_1,\ldots,q_k,z$, there exists a linear functional $g\colon \mathbf{F}^n \to \mathbf{F}$, such that, for any $x \in \{0,1\}^n$, $\mathbf{rec}_{\mathcal{P}}((a_1,\ldots,a_k),z) = g(x)$ where $a_j = \mathbf{ans}_{\mathcal{P}}(j,x,q_j)$. That is, even if the queries and the reconstruction information are badly formed, the reconstructed value is “well-behaved” in the sense that it is equal to some linear combination of the data. All known multi-database PIR schemes [10], [1], [9], [22] fit into this category.[17]

In the linear case, the goal of the testing procedure is to verify efficiently that the linear combination corresponding to the commodities c is correct, i.e., equal to x_r, while keeping r private from the databases. Note that to achieve *absolute* confidence in commodities’ correctness, the “black-box approach” requires that the tested strings span the linear space $\mathrm{GF}(q)^n$, implying that at least n data strings must be tested. However, settling for a small probability of one-sided error, a much more efficient solution to this problem can be obtained as a typical application of *small-bias probability spaces* [26]. The following fact is proved in [26].

Fact 2. *For any $n \in \mathcal{N}$ and $\epsilon > 0$, there is (an efficiently constructible) meta-test-set $\mathcal{T}_{n,\epsilon} \subseteq (\mathrm{GF}(q)^n)^l$, where $l = O(\log(1/\epsilon))$, such that*:

- *$|\mathcal{T}_{n,\epsilon}|$ is polynomial in n and $1/\epsilon$.*
- *For every $y \in \mathrm{GF}(q)^n$, $y \neq 0$, at most an ϵ-fraction of the test-tuples $(w_1, \ldots, w_l) \in \mathcal{T}_{n,\epsilon}$ satisfy $y \cdot w_b = 0$ for all $1 \leq b \leq l$.*

We now use Fact 2 to verify commodities with error probability ϵ:

1. The user picks a random index $d \in [|\mathcal{T}_{n,\epsilon}|]$ and sends it to each database.
2. Each database $\mathcal{DB}_j$ finds the dth test-tuple in $\mathcal{T}_{n,\epsilon}$, $(w_1, \ldots, w_l)$, and replies with $(a_j^1, \ldots, a_j^l)$, where $a_j^b = \mathbf{ans}_{\mathcal{P}}(j, w_b, q_j)$.
3. The user accepts if the commodities were correct on all l selected test strings; that is, if, for every $1 \leq b \leq l$, $\mathbf{rec}_{\mathcal{P}}((a_1^b, \ldots, a_k^b), z)$ is equal to the rth entry of w_b.

Since the random test index d is independent of the commodities, the above testing procedure does not compromise the user’s privacy. If the tested commodities are correct, the user always accepts. If they are incorrect, the user accepts with probability at most ϵ; to see this, note that if $v \neq e_r$ represents the linear combination corresponding to incorrect commodities, then $v - e_r \neq 0$, implying that with probability at least $1 - \epsilon$ there is a test vector w_b from the selected test-tuple such that $v \cdot w_b \neq e_r \cdot w_b$.

The communication complexity of the scheme is $O(\log n + \beta \log(1/\epsilon))$, where β is the answer complexity of $\mathcal{P}$. We note that one can use a simpler construction of $\mathcal{T}_{n,\epsilon}$ for constant ϵ (see [26]) and amplify success probability by independent repetitions, yielding a computationally easier procedure with a slightly worse asymptotic communication of $O((\log n + \beta) \log(1/\epsilon))$.

We finally remark that while our definition of commodity correctness does not directly apply to the polynomial interpolation scheme from Section 6 (as it is not composed of atomic schemes), it is possible to handle this scheme as well within the same linear framework as above.

[17] While known single-database schemes do not directly fit into this category, similar methods can be applied to the schemes from [23] and [30].

9. Extensions

In this section we discuss two extensions of the results presented in previous sections.

9.1. *Protecting Data Privacy*

Ordinary PIR schemes may reveal to the user (and especially to a dishonest user who does not follow his protocol) more information about the data string x than just a single data record x_i. This is a serious disadvantage in some scenarios, e.g., when the user is required to pay for each data item he retrieves. In [13] and [27] it is shown how to transform ordinary PIR schemes into stronger schemes, which protect the privacy of the data in the sense that *any* user, even a dishonest one, cannot learn more than a single data record in each invocation.

All the results of this work can be directly applied to such stronger schemes as well. Moreover, our use of PIR schemes with a very simple answer structure (see Section 3) makes it particularly easy to satisfy the extra data privacy requirement. In fact, the (moderate) overhead incurred by the transformations in [13] and [27] can be almost totally eliminated in the commodity-based setting, provided that not all servers collude with the user. For instance, consider commodity schemes in which the user reconstructs x_i by taking the exclusive-or of k answer bits $a_1, \ldots, a_k$. This is the case for the optimized version of the scheme $\mathcal{C}_{m,t,d}$ (see Corollary 3), as well as for commodity schemes constructed from the computational PIR scheme $\mathcal{P}_3$. If each answer bit a_j in these schemes is masked with a bit r_j such that the k bits $r_1, \ldots, r_k$ are random subject to the constraint that their exclusive-or is 0, then the only information about x revealed by the masked answers is a single data bit x_i. Letting the servers provide random masks r_j as above, we obtain commodity schemes that maintain data privacy with respect to an *honest* user, who sends the same shift amount Δ to all databases. A technique from [10] can be used to prevent a dishonest user, sending different shift amounts to different databases, from learning any information about the data. This requires only $O(\log n)$ additional commodity bits, and yields a commodity scheme which protects the privacy of the data against *any* (possibly dishonest) user.[18]

9.2. *Application to Private Information Storage*

Most of the results presented in this work can be adapted to the related problem of *Private Information Storage* introduced in [28]. Private information storage schemes allow a user to write and read data privately to/from a data string which is *secret-shared* (rather than replicated) among several databases. In the case of writing, this means that both the address of the written record and its contents should be hidden from each collusion of t databases.

In the following we only deal with 1-*round* storage schemes; this is contrasted with the main construction of [28], which requires logarithmically many rounds of interaction. We also assume that the "write" operation specifies an additive change to the ith

[18] Note that in the underlying PIR schemes the user has much more cheating power; by choosing invalid queries the user can learn linear combinations of large sets of data bits. In the commodity schemes the user's cheating is restricted to specifying different shift amounts, which can be more easily taken care of.

record, say over a finite field, rather than overwrite it with a specific value (e.g., in the case of single bit records, the user determines whether or not to flip the ith data bit). An "overwrite" operation can then be implemented using one read operation for retrieving the ith record, followed by one write operation to change (or unchange) its value.

More formally, a 1-round storage scheme is defined as follows. The "read" operation proceeds as in the case of PIR, except that the distributed representation of the data string is different (i.e., the data string is secret-shared rather than replicated). In a "write" operation, the user sends to each database a query string, called *command*, which is interpreted by each database to represent some transformation of its share of the data string. After performing these transformations, the shares held by the databases should represent an appropriately modified data string, in a way that will be consistent with subsequent read and write operations performed by *any* user. Such a scheme is said to be *t-private* if the commands viewed by any t databases give them no information (in the appropriate sense) on the write address or the change amount.

The notion of a *commodity storage scheme* can be defined in the natural way. We now argue that an analogue of the atomic commodity PIR schemes from Section 4 exists for storage schemes as well. Consider any 1-round k-database storage scheme in which the data string is shared recordwise (implying that shifting all shares by the same amount results in a valid representation of the shifted data string). For simplicity, we also assume that this storage scheme applies to single-bit data records, and that there is a dummy location which is not considered a part of the data string (so that to unchange the data the user may flip the dummy bit). A "write" operation in a corresponding commodity storage scheme can then proceed as follows:

- The server picks a random storage index $r \in Z_n$ and a k-tuple of commands for flipping x_r. Each command is sent to the corresponding database and the index r to the user.
- In the on-line stage, the user sends to each database a shift amount $\Delta = i - r \pmod{n}$. Each database: (1) cyclically shifts its share by Δ records to the left; (2) interprets its commodity command and performs the required transformation on the shifted share; and (3) shifts the transformed share back by Δ records to the right.

The read operation for the commodity scheme can be obtained from the original read operation as in atomic commodity PIR schemes.

Single-round storage schemes on which the above transformation may operate can be based on any of the PIR schemes $\mathcal{P}_1^k, \mathcal{P}_2^{t,d}$, and $\mathcal{P}_3$, with similar storage cost as the retrieval cost of the PIR schemes.[19] In fact, a 1-round storage scheme can be based on any PIR scheme in which the user's query is interpreted as asking for a *single* linear combination of the data records, over some finite field. In a corresponding storage scheme, the data string will be equal to the sum of all its shares; a write operation, adding 1 to x_i, is implemented by having each database add to its share the coefficient vector of the linear combination corresponding to a query pointing to x_i.

[19] If the "read" operation is implemented via a multi-database PIR scheme, the number of databases should be increased (as in [28]) to allow sufficient replication of each share.

Generally, attempting to construct storage schemes with higher server-privacy thresholds turns out to be more problematic, as the composition technique of Section 5 does not seem to be applicable in its full generality to the case of storage. However, it is possible to directly construct multiserver storage schemes with similar parameters to the commodity PIR schemes of Corollary 2 or Theorem 4.

Appendix A. The PIR Scheme $\mathcal{P}_4^d$

In this appendix we describe the PIR scheme $\mathcal{P}_4^d$ and some possible optimization of this scheme in a setting where a public source of randomness is available.

The scheme $\mathcal{P}_4^d$ can be obtained by a straightforward modification of the recursive construction from [23]. It uses the quadratic residuosity-based public-key encryption scheme from [20] described below. The public key is a κ-bit modulus $N = pq$, where p, q are two large primes satisfying $p \equiv q \equiv 3 \pmod 4$, and the private key is the pair (p, q). Let J_N^{+1} denote the multiplicative group of residues modulo N with Jacobi symbol 1. To encrypt a bit 0 we let $E_N(0) = r^2 \pmod N$, and to encrypt a bit 1 we let $E_N(1) = -r^2 \pmod N$ where r is a random residue modulo N. Note that an encryption of 0 (resp. 1) is a random quadratic residue (resp. quadratic nonresidue) in J_N^{+1}. If $m = m_1 m_2 \cdots m_\ell$ is a message of length ℓ, then m is encrypted by independently encrypting each of its ℓ bits; that is, $E_N(m) = E_N(m_1)E_N(m_2)\cdots E_N(m_\ell)$. The decryption function $D_{(p,q)}(c)$, where c is a $\kappa\ell$-bit ciphertext, proceeds by parsing c into κ-bit residues $(c_1, \ldots, c_\ell)$, and extracting the quadratic character of each residue using the private key (p, q).

The scheme $\mathcal{P}_4^d$ proceeds as follows. Assume that $n = a^d$ for some integer a. The user views the data string x as embedded in a d-dimensional cube of length a, and naturally identifies his retrieval index i with its coordinates $(i_1, \ldots, i_d) \in Z_a^d$. The user's query consists of the public key N, along with da independent encryptions $(c_0^1, \ldots, c_{a-1}^1), \ldots, (c_0^d, \ldots, c_{a-1}^d)$, where each $c_{a'}^{d'}$ is an encryption of the bit 1 if $i_{d'} = a'$ and an encryption of the bit 0 otherwise. In other words, the query includes an encryption of the d length-a unit vectors $e_{i_1}, \ldots, e_{i_d}$. The private key (p, q) is taken to be the reconstruction information. To specify how the database computes its answer, we define an operator **select**(y, c) as follows. Let y be an a-record data string with record length ℓ, and let c be an a-tuple of ciphertexts, where each ciphertext $c_{a'}$, $a' \in Z_a$, is an element of J_N^{+1}. We define **select**(y, c) to be a string of length $\kappa\ell$, obtained by concatenating the ℓ residues $s_0, s_1, \ldots, s_{\ell-1}$, where

$$s_{\ell'} = \prod_{a'=0}^{a-1} c_{a'}^{(y_{a'})_{\ell'}} \pmod N$$

(and each $s_{\ell'}$, $0 \le \ell' < \ell$, is represented using κ bits). Note that if c encodes a length-a unit vector $e_{i'}$ and y is a data string consisting of a records of length ℓ, then **select**(y, c) is a $\kappa\ell$-bit encoding of $y_{i'}$. The database's answer will be computed using d successive applications of the **select** operator, each having the effect of decreasing the dimension of the data cube by 1 and increasing the representation size of each entry by a factor of

κ. Specifically, the database starts by letting $y^{(d)} = x$, and for $d' = d-1, d-2, \ldots, 0$ lets $y^{(d')}$ be an $a^{d'}$-record data string, with record length $\ell^{(d')} = \kappa^{d-d'}$, defined by

$$y^{(d')}_{(i_1,i_2,\ldots,i_{d'})} = \mathbf{select}((c_0^{d'+1}, \ldots, c_{a-1}^{d'+1}), (y^{(d'+1)}_{(i_1,i_2,\ldots,i_{d'},0)}, \ldots, y^{(d'+1)}_{(i_1,i_2,\ldots,i_{d'},a-1)})).$$

The database replies with a data string $y^{(0)}$, whose single record may be intuitively viewed as a "d-level encoding" of x_i. Finally, the user reconstructs x_i from this answer by successively applying the decryption function $D_{(p,q)}$ to the answer d times.

In a setting where a public source of randomness is available, the query complexity of $\mathcal{P}_4^d$ may be improved by an asymptotic factor of κ (and in fact the same improvement applies to the query complexity of the original scheme from [23]). The idea is to modify the scheme $\mathcal{P}_4^d$ by first letting the user and the database parse the public random string as a sequence of random elements in J_N^{+1}, and then replacing each residue sent by the user with a single correction bit. Since (-1) is a quadratic nonresidue modulo any Blum integer N, the database can flip the quadratic character of any public residue simply by negating it. The query complexity of the modified scheme is $\kappa + dn^{1/d}$ (in opposition to $\kappa + d\kappa n^{1/d}$ of $\mathcal{P}_4^d$) and its answer complexity is κ^d (as of $\mathcal{P}_4^d$).

Appendix B. General Commodity Testing

In this section we address the general problem of testing commodities which correspond to an *arbitrary* PIR scheme. Note that in this case, achieving absolute confidence in commodity correctness using the "black-box approach" requires all 2^n data strings to be tested. Again, settling for a small probability of error one can do much better, using straightforward sampling techniques.

Informally, the testing procedures will either reject commodities or give statistical evidence that their correctness ratio is high. To avoid the worst case possibility of having a certified commodity fail on a specific data string x, the on-line retrieval protocol will be augmented to include randomization of the data string, as well as repeated querying for amplifying success probability.

We start by describing a procedure which is (statistically) secure against servers with unlimited computational power, but requires the use of a public random string picked independently of the commodities. This need for public randomness is dispensed with in what follows. For the sake of simplicity, we refer only to an *atomic* scheme $\mathcal{C}_{\mathcal{P}}$. The techniques can be adapted to any of our multiserver schemes as well.

COMMODITY STAGE:

1. The server, on input $1^\kappa, 1^n$, distributes commodities $c = ((r, z), (q_1, \ldots, q_k))$ as in $\mathcal{C}_{\mathcal{P}}(\kappa, n)$.
2. The user and the databases parse the public random string as $y_1, y_2, \ldots, y_\kappa$, where each y_d is n-bit long, and test the commodities c on each data string. That is, for each $1 \le d \le \kappa$, each database $\mathcal{DB}_j$ replies with $a_j = \mathbf{ans}_{\mathcal{P}}(j, y_d, q_j)$, and the user verifies that $\mathbf{rec}_{\mathcal{P}}((a_1, \ldots, a_k), z)$ is equal to the rth bit of y_d.
3. If the commodities fail the test, the user rejects. Otherwise, the user and the databases proceed to the retrieval stage.

RETRIEVAL STAGE:

1. The user and the databases parse the remainder of the public random string as $z_1, z_2, \ldots, z_\kappa$, where each z_d is n-bit long.
2. The user and the databases invoke the original retrieval scheme κ times, where in the dth invocation the databases replace the data string x by the (random) string $x \oplus z_d$.
3. The user reconstructs l answer bits $b_1, \ldots, b_\kappa$ according to the original reconstruction function, and outputs the majority vote of the bit values $b_d \oplus (z_d)_i$.

The second part of the next claim follows from a standard application of Chernoff bounds.

Claim 2. *The above testing procedure satisfies the following*:

- *If the commodities c are correct, the user will always output the correct data bit.*
- *For any commodities c, data string x, retrieval index i, and security parameter κ, if the user does not reject c at the commodity stage, then the probability that his output is wrong* (*i.e., is different from x_i*) *is $2^{-\Omega(\kappa)}$.*

We now argue that, under mild cryptographic assumptions, public randomness can be replaced by a shorter seed sent from the user to each database.

Claim 3. *Suppose there exists a nonuniformly secure*[20] *pseudorandom generator G: $\{0,1\}^{\kappa(L)} \to \{0,1\}^L$* (*i.e., any polynomial-size circuit family distinguishes U_L from $G(U_{\kappa(L)})$ with at most a negligible advantage in L, where U_ℓ is the uniform distribution on ℓ-bit strings*). *Let $L = 2\kappa n$ denote the total length of the public random string in the above testing procedure. Now modify the procedure by replacing the public random string with a random seed of size $\kappa(L)$ sent from the user to each database, so that both the user and the databases can apply G to the seed to obtain a common pseudorandom string of length L. Then the modified procedure satisfies the following*:

- *If all commodities are correct, the user will always output the correct data bit.*
- *For any commodities c, data string x, retrieval index i, and security parameter κ, if the user does not reject c at the commodity stage, then the probability that his output is wrong* (*i.e., is different from x_i*) *is negligible in κ.*

Proof. If (for infinitely many κ's) there exist $c_\kappa, x_\kappa, i_\kappa$ which make the user err with $\kappa^{-O(1)}$ probability, then a truly public random string of length $L \geq \kappa$ can be distinguished from a pseudorandom one with a $\kappa^{-O(1)}$ advantage, contradicting the pseudorandomness assumption. Specifically, the κth distinguishing circuit takes an $L = 2\kappa|x_\kappa|$ bit string as input, then simulates the above procedure (with $c_\kappa, x_\kappa, i_\kappa$) using its input as the public random string, and finally outputs 0 if the user's output is equal to x_i and 1 otherwise.

[20] Nonuniform security may be relaxed to uniform security if a corrupt server is restricted to be computationally efficient.

Since the algorithms $\mathbf{ans}_{\mathcal{P}}$, $\mathbf{rec}_{\mathcal{P}}$, and $\mathbf{que}_{\mathcal{C}_{\mathcal{P}}}$ are efficient, the size of such a circuit can be polynomial in L, the length of its input. $\square$

Substituting a "sufficiently secure" seed size for $\kappa(L)$ (e.g., $\kappa(L) = L^c$ for any $c > 0$ under standard cryptographic assumptions, or $\kappa(L) = polylog(L)$ under more ambitious assumptions), we get a communication efficient testing procedure for the general case (though not quite as efficient as for the linear case). We finally note that more general techniques for derandomizing BPP algorithms (see [16] for a survey) may be used to improve the above procedure.

References

[1] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Proc. of* 24*th ICALP*, 1997.

[2] D. Beaver. Commodity-based cryptography. In *Proc. of* 29*th STOC*, pages 446–455, 1997.

[3] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proc. of* 7*th STACS*, pages 37–48, 1990.

[4] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Proc. of CRYPTO*, 1990.

[5] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single database private information retrieval. In *Proc. of* 31*st STOC*, pages 89–98, 1999.

[6] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of* 20*th STOC*, pages 1–10, 1988.

[7] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Proc. of EUROCRYPT* '99, pages 402–411, 1999.

[8] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of* 20*th STOC*, pages 11–19, 1988.

[9] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of* 29*th STOC*, pages 304–313, 1997.

[10] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of* 36*th FOCS*, pages 41–50, 1995.

[11] G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrival implies oblivious transfer. In *Proc. of EUROCRYPT* '00, 2000.

[12] Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. In *Proc. of* 2*nd RANDOM* 98, 1998.

[13] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information schemes. In *Proc. of* 30*th STOC*, pages 151–160, 1998.

[14] O. Goldreich. *Foundations of Cryptography* (*fragments of a book*). Electronic Colloquium on Computational Complexity, 1995. Electronic publication: http://www.eccc.uni-trier.de/eccc-local/ECCC-Books/eccc-books.html.

[15] O. Goldreich. On the foundations of modern cryptography. In *Proc. of CRYPTO* '97, pages 46–74, 1997.

[16] O. Goldreich. *Modern Cryptography, Probabilistic Methods and Pseudo-Randomness*. Springer-Verlag, New York, 1999.

[17] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). In *Proc. of* 19*th STOC*, pages 218–229, 1987.

[18] S. Goldwasser. Multi-party computations: past and present. In *Proc. of* 16*th PODC*, pages 1–6, 1997.

[19] S. Goldwasser. New directions in cryptography: twenty some years later. In *Proc. of* 38*th FOCS*, pages 314–324, 1997.

[20] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28:270–299, 1984.

[21] J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[22] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval. In *Proc. of* 31*st STOC*, pages 79–88, 1999.
[23] E. Kushilevitz and R. Ostrovsky. Single-database computationally private information retrieval. In *Proc. of* 38*th FOCS*, 1997.
[24] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for single-server private information retrieval. In *Proc. of EUROCRYPT* '00, 2000.
[25] F.J. Macwilliams and N.J. Sloane. *The Theory of Error Correcting Codes*. North-Holland, Amsterdam, 1978.
[26] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proc. of* 22*th STOC*, pages 213–223, 1990.
[27] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of* 31*st STOC*, pages 245–254, 1999.
[28] R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of* 29*th STOC*, pages 294–303, 1997.
[29] A. Shamir. How to share a secret. *Communications of the ACM*, 22(6):612–613, June 1979.
[30] J.P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Proc. of ASIACRYPT* '98, pages 357–371, 1998.
[31] A.C. Yao. Protocols for secure computations. In *Proc. of* 23*rd FOCS*, pages 160–164, 1982.