Universal Service-Providers for Database Private Information Retrieval

Giovanni Di-Crescenzo[*] Yuval Ishai [†] Rafail Ostrovsky[‡]

February 22, 1998

Abstract

We consider the question of private information retrieval in the so-called "commodity-based" model. This model was recently proposed by Beaver for practically-oriented service-provider internet applications. In this paper, we show the following, somewhat surprising, results regarding this model for the problem of private-information retrieval: (1) the service-provider model allows to dramatically reduce the overall communication involving the user, using off-line pre-processing messages from "service-providers" to databases, where the service-providers do not need to know the database contents, nor the future user's requests; (2) our service-provider solutions are resilient against more than a majority (in fact, all-but-one) coalitions of service-providers; and (3) these results hold for *both* the computational and the information-theoretic setting.

More specifically, we exhibit a service-provider algorithm which can "sell" (i.e., generate and send) "commodities" to users and databases, that subsequently allow users to retrieve database contents in a way which hides from the database which particular item the user retrieves. The service-providers need not know anything about the contents of the databases nor the nature of the user's requests in order to generate commodities. Our commodity-based solution significantly improves communication complexity of the users (i.e., counting both the size of commodities bought by the user from the service-providers and the subsequent communication with the databases) compared to all previously known on-line private information retrieval protocols (i.e., without the help of the service-providers). Moreover, we show how commodities from different service-providers can be *combined* in such a way that even if "all-but-one" of the service-providers collude with the database, the user's privacy remains intact. Finally, we show how to re-use commodities in case of multiple requests (i.e., in the amortized sense), how to "check" commodity-correctness, and how some of the solutions can be extended to the related problem of *Private Information Storage*.

[*]CSE department, UCSD, La Jolla, CA, USA. E-mail: giovanni@cs.ucsd.edu. Part of this work was done while visiting Bellcore.

[†]Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: yuvali@cs.technion.ac.il. Part of this work was done while visiting Bellcore.

[‡]Bell Communication Research, Morristown, NJ, USA. E-mail: rafail@bellcore.com.

1 Introduction

Cryptography in the 90's: With the wide-spread use of World-Wide Web and internet applications, cryptographic protocols are increasingly used in commercial web-based settings. Hence, while the trend of the 1980's was to establish general plausibility results, the trend of the 1990's is to consider solutions which are *both* provably secure and efficiently implementable in practical applications (for more general discussion on this topic see surveys of Goldreich [10] and Goldwasser [13, 12]). The current trend of developing practical and provably secure solutions is not only of considerable practical importance, but also poses interesting theoretical questions, since it often requires to consider different, more practically-oriented models and to devise efficient and secure solutions for these models. This is the case of the current paper as well — here, we consider internet "service-provider" model, introduced by Beaver [2], and show how with the help of service-providers we 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!

Service-provider model: Motivated by a "client-server" paradigm, Beaver [2] proposed a new "service-provider" model, where there are several "servers" which off-line "sell commodities" to users, who can use these commodities to perform more cheaply various cryptographic protocols. The reason this model is of interest is that current internet applications are based on the client-server approach, which allows service-providers to send messages to users, but does not allow users to interact with service providers. We remark that this setting is clearly much more restrictive than secure multi-party protocols (such as [21, 11, 5, 6]) that require inefficient point-to-point multi-round protocols. An advantage of the service-provider model is that service-providers do not need to know private inputs of users since they are restricted to send a single message (called "commodity") to each user. In [2], Beaver showed how to achieve so-called "One-Out-of-Two Oblivious Transfer" and "Multicast" in this model, provided that the majority of the servers are honest. We consider this model in the practically-oriented setting of remote database information retrieval.

Private Information Retrieval: *Private information retrieval* schemes allow a user to retrieve information from a remote database (or several non-communicating copies of the database) in such a way that the database does not get any information about the user's query. Formally, the database is viewed as an n-bit string x out of which the user wishes to retrieve the i-th bit x_i, while maintaining i private. Obviously, if the database sends its entire contents x to the user this will hide which particular address i the user is interested in, but the *communication complexity* of such naive solution is prohibitive. (For example, consider a database of all U.S. patents, where user is interested in retrieving some patent without revealing to the database which patent is of interest to him.) Hence, the main research goal in private information retrieval protocols considered in the last several years is to reduce the communication complexity. We now briefly mention some of the work done in this area in the past. Let k we denote the number of non-communicating copies of the database. Private information retrieval with information-theoretic user privacy was introduced by Chor, Goldreich, Kushilevitz and Sudan [8], who give schemes with communication complexity of $O(n^{1/3})$ bits for $k = 2$ databases, $O(n^{1/k})$ bits for $k \geq 3$ 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 $O(n^{1/(2k-1)})$. *Computational* Private Information Retrieval schemes (i.e., schemes where the privacy is only with respect to polynomial-time databases, relying on certain intractability assumptions) were considered in Chor and Gilboa [7] and

Ostrovsky and Shoup [19] for $k \geq 2$ databases, and for the case of a single database by Kushilevitz and Ostrovsky [15]; for any $\varepsilon > 0$ such schemes can be implemented with communication complexity $O(n^{\varepsilon})$.

SERVICE-PROVIDER PRIVATE INFORMATION RETRIEVAL: Informally, the setting is as follows. Similarly to private information retrieval, there is a *user* with private input (address) i and one or more *databases* holding copies of an n-bit data string x. As before, the user wishes to retrieve x_i while not revealing i to the databases. Additionally, there are one or more *commodity servers*, running identical probabilistic polynomial time sampling algorithm to generate "commodities". That is, a *commodity scheme* consists of the following stages:

1. (off-line *commodity stage*): First, all servers on input $1^{|n|}$ (and an optional security parameter) run identical probabilistic polynomial time sampling algorithm which generates (private) individual messages (which we call *commodities*) for the user and the databases. We note that servers never interact with one another, do not know database contents, do not know the future user's requests and do not even know if there are other servers in the system, and if so, how many other servers there are.
2. (on-line *retrieval stage*): With commodities from pre-processing stage as private inputs, the user and the databases execute some private information retrieval protocol in which the user sends queries to the databases and receives answers in return.

Note that in case of more than one server, servers run the same sampling algorithm. In a practical setting, we envision many (perhaps competing) servers, where the user decides how many of them to use. We measure both the communication in the off-line commodity stage (i.e., size of commodities) and in the on-line retrieval stage (i.e., communication between the user and the databases). Our main objective is to minimize the total communication complexity for the users (in both stages), and shift more expensive communication to off-line pre-processing messages from servers to databases. It should be noted that one cannot expect to have better *total* communication than in ordinary private information retrieval schemes (i.e. without servers), since the servers can be simulated by the user to obtain ordinary schemes of the same total complexity). We stress though that all our schemes allow minimizing the total communication involving the *user* to be logarithmic in n, and polynomial in the security parameter (in case of single-database computationally private schemes).

Another major goal is to guarantee the user's privacy even when some of the servers dishonestly collude with databases. It is important to remark why we consider collusions between servers and a database to be a bigger threat than collusions between databases. First, in single-database computational schemes, the latter problem does not exist at all. Second, commodity servers are arguably more likely to be corrupt (since they do not hold any confidential data, there are potentially many of them, etc.). Finally, a server with so-called "honest-but-stupid" faults [2] (e.g., one with a bad random number generator) may cause the same damage as a server which colludes with the databases. (In contrast, honest faulty databases do not compromise the user's privacy, neither in our schemes nor in previously existing private information retrieval schemes.) In most of our multiple server schemes, even if all-but-one of the servers collaborate with database(s), the user's privacy still remains intact.

OUR RESULTS: We start by presenting a single-server scheme, where as long as this server does not collude with the databases, the user's privacy is guaranteed. We then show how to *combine* commodities so that the user's privacy is guaranteed even if all-but-one of the servers collaborate with a database.

In particular, by establishing general transformations from PIR schemes to commodity schemes (and by "plugging in" appropriate modifications of PIR schemes from [8, 7, 15]) we construct the following commodity schemes:

- **Computational single database case:** For any constants $m \geq 1$ and $\varepsilon > 0$ we construct an m-server, single-database computational scheme, withstanding collusions of the database with up to $m-1$ servers, with user's communication of size $O(\log n + poly(K))$ (counting both the user's commodity and on-line communication) and server-database commodity complexity $O(K \cdot n^{\varepsilon})$ (where K is a security parameter and security is based on the Quadratic Residuosity Assumption).
- **Computational multiple-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 $K \cdot 2^{O(\sqrt{\log n})}$ (assuming the existence of one-way functions). Schemes of this type are superior to the others when the server-privacy threshold is small and the database size is large.
- **Information-theoretic multiple-database case:** For any constant integers $m, t, d \geq 1$, we construct an m-server, $(mtd+1)$-database information-theoretic scheme, withstanding collusions of up to t databases and $m-1$ servers, with user's communication of size $O(\log n)$ and server-database commodity complexity $O(n^{1/d})$. Schemes of this type are superior to the others when the database size is relatively small.

We then proceed to show how to make amortized cost of service-provider solutions even cheaper and how to test commodities:

- **Amortizing commodity cost for multiple queries**: In all of our s-private schemes (i.e., those that can withstand s dishonest servers), by using $m > s+1$ servers the amortized server-database commodity cost per query can be reduced to $\frac{1}{s+1} \cdot \frac{m}{m-s}$ times the cost of a single query in the $(s+1)$-server scheme (while maintaining s-privacy). If one is willing to assume that the databases have no *a-priori* knowledge of the user's interest patterns, much better amortization can be achieved.
- **Commodity testing**: We give procedures for verifying validity of commodities supplied by the servers. This allows to ensure correctness of our schemes even in the presence of faulty or dishonest servers (and should not be confused with the user's privacy, which in all our schemes is protected from up to s such servers colluding with up to t databases). This problem is particularly natural in our setting, where some of the (potentially many) servers are likely to be malfunctioning. Moreover, the testing procedure can be carried out off-line, after the distribution of commodities and before the actual retrieval.

Another potentially useful side-effect of the service-providers model is the ability of servers to function as *retrieval escrow* agents. That is, a sufficiently large coalition of servers may be granted the legal right to recover the user's retrieval index, given a transcript of the user's query. In our $(m-1)$-private m-server schemes, a coalition of *all* servers with a database is required to reveal the user's index. It is possible to generalize this to schemes in which smaller coalitions can recover the user's index (while sufficiently small coalitions can still learn nothing about it), with very little overhead.

Finally, we show that some of the server-based solutions can be applied to the related problem of *Private Information Storage*.

Our work has implications both for server-based model and for the private information retrieval. We list some of them below.

BENEFITS OF OUR SOLUTIONS FOR PRIVATE INFORMATION RETRIEVAL: Our server-based schemes have the following advantages over ordinary private information retrieval schemes:

- Little on-line communication between users and databases, and little total communication involving the user. The overall communication, including all commodities, is comparable to that of the original private information retrieval schemes.
- The user's privacy does not depend on the quality of its randomness (in fact, in most of our schemes the user is deterministic).
- As a side-effect, protecting data privacy against a potentially malicious user (in a sense that the user does not get more information than the single entry he has "paid for") is made much easier assuming that the servers are honest. In fact, all of our schemes can be modified to protect data privacy with almost no complexity overhead.
- Since the commodities do not depend on the data, even if the data string changes, the commodities can still be used; thus, commodities never expire.

COMPARISON WITH PREVIOUS SERVER-BASED WORK: The only previous work explicitly dealing with this server-based model is last year's Commodity Based Cryptography paper of Beaver [2]. It is instructive to illuminate some points of comparison between this work and the previous one:

- 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.
- Our solutions achieve resilience to collusions of databases with up to $m-1$ servers (in oppose an optimal threshold of $(m-1)/2$ servers in Beaver's Oblivious-Transfer protocol). This better privacy level is made possible here because of the different setting, which allows either replication of data or computational privacy.
- In Beaver's protocols all servers function identically and independently. Moreover, the communication between servers to their clients is only one-way. All this holds for this work as well.

TECHNIQUES USED: At a first glance, it is not at all clear how a server who does not know database contents nor the future user's request can help in private information retrieval. Our first step is a *reduction* to standard private information retrieval: we show how a private information retrieval request to a *random* database position r can be used to prepare "generic" question for databases (with just the index r told to the user) which later on can be very cheaply converted by a user into his own arbitrary desired retrieval position i. We then introduce a new technique for combining these generic commodities, both in the single-database and the multi-database setting. Extending techniques from [8], we also give a direct solution for the information-theoretic multi-database setting, based on the method of low-degree polynomial interpolation. Finally, we introduce two additional algorithmic tools: one is a technique of amortizing the cost of the commodities and another for commodity testing.

ORGANIZATION: Section 2 contains some notation, as well as definitions of the private information retrieval (abbreviated as PIR) and commodity PIR models. In Section 3 we summarize the complexity parameters of specific PIR schemes which will be utilized in obtaining communication efficient commodity PIR schemes. In Section 4 we informally describe the main commodity schemes which are formally defined in subsequent sections. Section 5 introduces atomic commodity schemes, and Section 6 deals with composing them to improve their privacy properties. In Section 7 we construct multi-database commodity schemes based on polynomial interpolation. In Section 8 we show that the commodity cost of our schemes can be amortized over multiple queries. Section 9 provides procedures for testing the correctness of commodities distributed by the servers. Section 10 discusses extensions of the original problem; one of these extensions concerns the issue of protecting privacy of the data against the user, and another concerns extensions of the results to the problem of private information storage. Finally, the appendix contains some more detail on the PIR schemes summarized in Section 3, as well as more general commodity testing procedures.

2 Notation and Definitions

The following notations and conventions are used throughout. Z_n denotes the additive group of residues modulo n, and $[k]$ denotes the set $\{1, 2, \ldots, k\}$. By $\log n$ we denote $\log_2 n$, and by e_r, $r \in [n]$, we denote the r-th unit vector of length n. By default, 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. By $e \leftarrow \$(D)$ we denote a choice of an element e from a distribution D (or uniformly from a finite set D).

The following notation and definitions are specifically related to PIR and commodity-based PIR schemes. We let k denote the number of *databases*, an instance of which is denoted $\mathcal{DB}_j$, and let m denote the number of *commodity servers* (or servers for short), an instance of which is denoted $\mathcal{S}_h$. By x we denote a *data string*, consisting of n ℓ-bit records ($\ell = 1$ by default), which is held by all k databases and is unknown to the user and the servers. The position, also called *index*, of a data record which the user wants to retrieve is denoted by i, where $i \in Z_n$. In the computational setting, K will denote a security parameter.

Notation and definitions for standard PIR Schemes

A *PIR* scheme is a randomized protocol, in which the user sends a *query* to each database, and receives an *answer* in return.[1] At the end of the interaction, the user applies some *reconstruction* function to the answers, the index i and its private coin tosses, to obtain the desired data record x_i.

Fixing the parameters n, ℓ (or K, n, ℓ in the computational case), a PIR scheme $\mathcal{P}$ is formally defined using the following notation:

- $\text{rand}_{\mathcal{P}}$ denotes the distribution from which the user's random input is chosen (e.g., a uniform distribution on all binary strings of some fixed length);
- $\text{query}_{\mathcal{P}}(i, \sigma)$ denotes the k-tuple of queries generated by the user on retrieval index i and random input σ;
- $\text{reconstruct-info}_{\mathcal{P}}(i, \sigma)$ extracts from the user's random input σ and the retrieval index i a reconstruction information string z, such that reconstruction can later depend on the answers, i and z alone, without depending on σ. Although $z = \sigma$ will always do, it turns out that a much shorter z can be used in all currently known PIR schemes,

[1] 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.

without effecting the computational efficiency of reconstruction.[2] This feature, which is not very useful in the original PIR setting, turns out to be important in our context.

- $\text{answer}^j_{\mathcal{P}}(x, q)$ denotes the answer of database $\mathcal{DB}_j$ on data string x and query q;
- $\text{reconstruct}_{\mathcal{P}}(a_1 \ldots a_k, i, z)$ denotes ℓ-bit record output by the user on answers $a_1, \ldots, a_k$, retrieval index i, and reconstruction information z.

In the computational setting, all the above functions must be computable in polynomial time (uniformly) in K, n, ℓ.

A PIR scheme is *correct*, if for any retrieval index i, user's random input σ and data string x, the reconstructed record is equal to x_i (assuming the databases are honest). Informally, a k-database PIR scheme is *t-private*, if any coalition of t databases cannot learn the retrieval index i from its view of the user's queries. Formal definitions of privacy in both the computational and information-theoretic setting are given below.

INFORMATION THEORETIC PRIVACY: A *t-private* information-theoretic PIR scheme must satisfy the following privacy requirement: under any two indices i, i', the communication seen (jointly) by any t databases is identically distributed.

COMPUTATIONAL PRIVACY: The information-theoretic privacy requirement is relaxed to computational indistinguishability, parameterized by a security parameter K, in the case of *computational PIR*. Formally, let $\mathcal{F}$ be a class of functions $f : \mathcal{N} \to \mathcal{N}$, where $\mathcal{F}$ is polynomially-closed (i.e., $f(K) \in \mathcal{F}$ implies that $p(f(K)) \in \mathcal{F}$ for any polynomial p). Such $\mathcal{F}$ will represent a bound on the adversary's resources as a function of the security parameter K. We say that a family of PIR schemes $\mathcal{P}(K, n, \ell)$ is *computationally t-private* with respect to $\mathcal{F}$, if the following condition holds: For any constant $c > 0$ and function $f \in \mathcal{F}$, there exists K_0, such that for any data size parameters n, ℓ, retrieval indices $i_1, i_2 \in Z_n$, collusion $T \subseteq [k]$ of t databases, security parameter $K \geq K_0$, and circuit C of size $f(K)$:

$$|\mathbf{Pr}_\sigma [C(Q_T(i_1, \sigma)) = 1] - \mathbf{Pr}_\sigma [C(Q_T(i_2, \sigma)) = 1]| < K^{-c},$$

where $Q_T(i, \sigma)$ denotes restriction of the k-tuple of queries $\text{query}_{\mathcal{P}(K,n,\ell)}(i, \sigma)$ to those viewed by databases in T. Finally, the adversary class $\mathcal{F}$ will usually be omitted, under the implicit understanding that it corresponds to the strength of an underlying intractability assumption. By default, $\mathcal{F}$ is taken to be the class of all polynomials, corresponding to the usual security assumptions which limit the adversary's power to be polynomial in the security parameter.

We note that this definition implies privacy in a sense similar to the definition in [7], where n serves both as a data size parameter and as a security parameter. Specifically, if the family $\mathcal{P}(K, n)$ is private under this two-parameter definition with respect to the default $\mathcal{F}$, then for any $\varepsilon > 0$ the family $\mathcal{P}'(n) \stackrel{\text{def}}{=} \mathcal{P}(n^\varepsilon, n)$ is private under the single-parameter definition. However, if $\mathcal{P}(K, n)$ is private with respect to a stronger class $\mathcal{F}$, then smaller functions of n can be substituted for K, as small as $polylog(n)$ in an extreme case (e.g., when $\mathcal{F} = 2^{O(n^\varepsilon)}$).

[2] In all information-theoretic schemes known to date [8, 1], as well as in the computational scheme of [7], no such reconstruction information is needed at all. In trapdoor-based computational schemes ([15, 17]), z should reflect the trapdoor information required for efficient reconstruction. For instance, $|z| = K$ suffices for the single-database scheme of [15] (since knowing the factorization of the K-bit modulus N is sufficient for efficient reconstruction).

Commodity-Based PIR Schemes

A *commodity-based PIR scheme* (or *commodity scheme* for short) consists of the following two stages.

Commodity-distribution stage: Each server $\mathcal{S}_h$ randomly and *independently of all other servers* creates a commodity C_h^u, sent to the user, and commodities $C_h^{db_1}, \ldots, C_h^{db_k}$, each sent to the corresponding database.

Retrieval stage: The user and the databases engage in a retrieval protocol, as in the original PIR setting, except that queries, answers and reconstruction may also depend on commodities.

A commodity scheme is *correct*, if at the end of the retrieval stage the reconstructed value is equal to x_i (assuming all parties are honest); such scheme is said to be (s,t)*-private*, if i is kept private from any collusion of at most s servers and t databases (which may be possibly dishonest). The formal definition of privacy in both settings (information-theoretic and computational) is similar to the analogous PIR definitions, except that in this case the view of (s,t)-collusions includes *commodities*, in addition to queries sent by the user. Since the default setting considered in other PIR works is $t=1$, "s-private" will stand for $(s,1)$-private.

Complexity

Complexity is measured, by default, in terms of communication. The *communication complexity* of a PIR scheme or commodity scheme is denoted (α,β), where α (denoted *query complexity*) is the maximal number of query bits sent from the user to any database, and β (denoted *answer complexity*) is the maximal number of answer bits sent from any database to the user. The *reconstruction information complexity* of a PIR scheme, denoted γ, is the maximal size of reconstruct-info(i,σ).

In the case of commodity schemes, 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 we always refer to infinite families of schemes, parameterized by the number of records n, a security parameter K (in the computational case) and the record size ℓ, the complexity measures $\alpha,\beta,\gamma,\delta^u,\delta^{db}$ may depend on these parameters.

Whenever the parameter ℓ is omitted, it is understood to be equal to 1. For instance, $\beta(K,n)$ will be used to denote the answer complexity of a computational scheme, on an n-bit data string and with security parameter K.

Block retrieval. In [8], communication balancing techniques are employed to reduce the communication cost of retrieving *blocks* of data, i.e. multi-bit data records. These techniques will not be useful for our purposes.[3] Instead, we implement retrieval of ℓ-bit records in the following "naive" way. Let x^w, $1 \le w \le \ell$, denote the n-bit string obtained by taking only the w-th bit from each record. To retrieve an ℓ-bit record: (1) the user sends a query as in the scheme for single-bit records; (2) each database answers the user's query ℓ times, once under each n-bit data string x^w; and (3) the user applies the original reconstruction function ℓ times, once for each answer. Hence, if the communication complexity of a PIR scheme

[3] The PIR schemes we will use are optimized to have the smallest answer complexity possible; in this setting, block retrieval techniques from [8] do not yield any improvement.

$\mathcal{P}(K,n)$ for retrieving a single bit is (α,β), then the induced block scheme $\mathcal{P}(K,n,\ell)$ is of complexity $(\alpha,\ell\cdot\beta)$. In the following we will freely use any PIR scheme $\mathcal{P}$ on data strings of arbitrary record size.

3 PIR Schemes with Low Answer Complexity

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

The following table summarizes the parameters of specific PIR schemes whose answer complexity is minimized to either K bits, in the computational single-database case, or a single bit, in the multi-database case. The parameters of some of these schemes will be explicitly referred to in the sequel. The parameter d appearing in the table can be substituted by any positive integer (including 1), and ε by any fraction $1/d$. In the "security type" column, "QRA" stands for the Quadratic Residuosity Assumption (cf. [14]), and "1-way" stands for the assumption of existence of 1-way functions.

name	DBs	privacy	α	β	γ	security type
$\mathcal{P}_1^{\varepsilon}$	1	1	$K+(1/\varepsilon)Kn^{\varepsilon}$	$K^{1/\varepsilon}$	K	computational (QRA)
$\mathcal{P}_1^{\varepsilon\,\prime}$	1	1	$K+(1/\varepsilon)n^{\varepsilon}$	$K^{1/\varepsilon}$	K	computational (QRA) (assuming public randomness)
$\mathcal{P}_2^{k}$	k	$k-1$	n	1	0	information theoretic
$\mathcal{P}_3$	2	1	$K\cdot 2^{O(\sqrt{\log n})}$	1	0	computational (1-way)
$\mathcal{P}_4^{t,d}$	$td+1$	t	$O(n^{1/d})$	1	0	information theoretic

The scheme $\mathcal{P}_2^k$ 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 $x\cdot q_j$, and the user reconstructs x_i by computing the exclusive-or of the k answer bits. (This is a simple generalization of an elementary scheme from [8]).

The other schemes are variants of schemes from [8],[7] and [15]. We refer the reader to Appendix A for descriptions of the required modifications.

Remark: The definition of computational privacy is insensitive to polynomial variation in the security parameter K (under the default adversary class $\mathcal{F}$); for instance, any computational scheme with answer complexity K can be regarded as a scheme with answer complexity $K^{1/100}$. However, the parameter K in the above table reflects the size of instances to the underlying computational problem which are used in the scheme (e.g., size of the modulus N in quadratic-residuosity based schemes). In practice, substituting for K a minimal instance size which is believed to be "sufficiently hard" for this computational problem can give a good idea of the actual cost of implementing the schemes.

4 Informal Overview

In this section we informally describe the main constructions of commodity schemes. For the sake of simplicity, we will usually refer only to *single* database schemes; the more general case is treated in the technical sections.

Atomic commodity schemes

Consider any 1-round single-database (computational) PIR scheme $\mathcal{P}$. Such scheme may be viewed as the following three-stage procedure: (1) the user computes a randomized query q pointing to the i-th data record; (2) the database computes an answer to this query based on the database contents; and (3) the user reconstructs the i-th data record, x_i, from the answer and some extra trapdoor information generated along with the query. While the communication and computation cost of each such step may vary from one scheme to another, none of the known PIR schemes is satisfactorily efficient in either of these aspects. The following simple idea allows to shift almost all the communication cost and a substantial part of the computation cost 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 r, along with its associated trapdoor information;
- Send the index r along with the trapdoor information 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 its desired 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 trapdoor 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 will be referred to as the *atomic commodity scheme* based on $\mathcal{P}$, and will be denoted $\mathcal{C}_{\mathcal{P}}$. The total communication involving the user, counting both the off-line commodity 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.

Composed commodity schemes

A major drawback of any atomic commodity scheme is the total dependency of the user's privacy on a proper behavior of the single server. Indeed, it suffices for the database to learn the index r picked by the server to recover the user's retrieval index i. Thus, even if the server uses a faulty source of randomness, let alone dishonestly cooperates with the database, the user's privacy is compromised. The latter problem can be alleviated by distributing the user's trust among several servers rather than one; we achieve this by composing atomic commodity schemes into multi-server schemes with improved privacy properties. Consider, for instance, two atomic commodity schemes: $\mathcal{C}_{\mathcal{P}_1}$ with server S_1, and $\mathcal{C}_{\mathcal{P}_2}$ with server S_2. The composed scheme will proceed as follows:

Commodity stage: 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 S_1, S_2, and q_1, q_2 denote the corresponding queries.

Retrieval stage: 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 e-th record of x', $0 \leq e < n$, will consist of the answer according to $\mathcal{P}_1$ to the commodity-query q_1 on the original data string x shifted by e places to the left. The retrieval of the i-th record of x can now be reduced to retrieval of the Δ-th record of x', where $\Delta = i - r_1 \pmod{n}$ is the query used in $\mathcal{C}_{\mathcal{P}_1}$ for retrieving the i-th record of x. The user retrieves this Δ-th record of x' using the retrieval procedure of $\mathcal{C}_{\mathcal{P}_2}$, based on commodities supplied by S_2. From this record, the user can apply the reconstruction procedure of $\mathcal{C}_{P_1}$ 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 effect the database's answer, whose size will be proportional to the this record size.

The query sent by the user in the composed scheme is $i - r_1 - r_2$. Since both r_1, r_2 are hidden from the database and exactly one of them is hidden from each server, i is kept hidden 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 S_1 but is unknown to the database and 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 above can be viewed as follows:

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

Using this presentation, it is easy to generalize the two-server composed scheme into an m-server scheme, which keeps i private from any collusion of the database and $m-1$ servers. In such 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 each oblivious shift increases the record size of the virtual data string by a multiplicative factor which is equal to the answer size of the underlying PIR scheme. Thus, for large values of m this approach will yield schemes with unrealistically large answer complexity. This problem can be avoided in the multiple-database case, which will be discussed in the sequel.

Multi-database schemes

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). Another advantage is their computational requirements, which are typically more modest. Finally, and in our context most importantly, they can potentially have the smallest answer complexity possible — as low as a single bit. In contrast, it is not hard to observe that a very low answer complexity implies poor level of computational privacy in single-database PIR schemes. Since the bottleneck of the previous multi-server 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.

One approach for utilizing multi-database schemes in the commodity setting is by a straightforward generalization of the previous solutions. A technical difficulty encountered in such generalization is that the number of databases has to grow exponentially with the server privacy threshold, to allow sufficient replication of the virtual data strings created by the database. Such multi-database schemes are discussed in Section 6. A direct approach, which allows to avoid this exponential growth, uses the technique of low-degree polynomial interpolation (extending similar solutions in the usual PIR setting). The idea is to have each server S_h share its chosen shift amount r_h among the databases, in a way that will keep the value of r_h private on one hand, and on the other hand will allow each database to perform some local computations with the combined effect of shifting a shared representation of x by Σr_h places. If this local computation can consist of a low-degree polynomial evaluation, then a small number of databases will suffice for reconstructing the desired record of the shifted data string. The exact implementation details are described in Section 7.

5 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 PIR scheme. Single-server schemes obtained via this transformation will be referred to as *atomic schemes*, and will subsequently be composed into schemes with improved privacy properties.

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

Proof. The atomic commodity scheme $\mathcal{C}_{\mathcal{P}}$ obtained from the PIR scheme $\mathcal{P}$ proceeds as follows. The server randomly chooses a retrieval index r, and then computes random queries $(q_1, \ldots, q_k)$ pointing to r, as specified by $\mathcal{P}$, along with reconstruction information z (if needed). The user's commodity consists of the pair (r, z), and $\mathcal{DB}_j$'s commodity is the corresponding query q_j. At the retrieval stage, the user computes the offset $\Delta \stackrel{\text{def}}{=} i - r \pmod n$, and sends it to all databases; each database $\mathcal{DB}_j$ replies with the answer (according to $\mathcal{P}$) to the commodity query q_j on a virtual data string obtained from cyclically shifting x by Δ places to the left. Finally, the user reconstructs x_i by applying the reconstruction function of $\mathcal{P}$ to the answers, the index r, and the reconstruction information z. This atomic commodity scheme is formally described in Figure 1.

Correctness follows from observing that when cyclically shifting x by Δ places to the left, the desired bit x_i moves to position $i - \Delta = r$, to which the commodity queries point. It remains to show that $\mathcal{C}_{\mathcal{P}}$ is private with respect to any coalition of t databases. Intuitively, the t-privacy of $\mathcal{P}$ ensures that the joint commodities distributed to any t databases give them no information on r (either computationally or information-theoretically, depending on the security type of $\mathcal{P}$), implying that such commodities along with the user's query $i - r$ keep i private.

Atomic-Scheme-$\mathcal{C}_\mathcal{P}$ /* $\mathcal{P}$: k-database PIR scheme */

Commodities :

1. $r \leftarrow \$(Z_n)$;
   $\sigma \leftarrow \$(\mathrm{rand}_\mathcal{P})$;
   $(q_1, q_2, \ldots, q_k) \leftarrow \mathrm{query}_\mathcal{P}(r, \sigma)$;
   $z \leftarrow \text{reconstruct-info}_\mathcal{P}(r, \sigma)$.
2. $C^u \leftarrow (r, z)$;
   $C^{db_j} \leftarrow q_j, \quad j = 1..k$.

Retrieval :

1. $\mathcal{U} \rightarrow \mathcal{DB}_j$: $\Delta \stackrel{\mathrm{def}}{=} i - r \pmod n$.
2. $\mathcal{DB}_j \rightarrow \mathcal{U}$: $a_j \stackrel{\mathrm{def}}{=} \mathrm{answer}^j_\mathcal{P}(x << \Delta, q_j)$
   (where $x << \Delta$ denotes a cyclic shift of x by Δ places to the left).
3. $\mathcal{U}$: Outputs $x_i = \mathrm{reconstruct}_\mathcal{P}(a_1 \ldots a_k, r, z)$.

Figure 1: The atomic single-server commodity scheme $\mathcal{C}_\mathcal{P}$.

For a formal proof of privacy we consider here the computational case; the information-theoretic case is simpler and is thus omitted. Let $T \subseteq [k]$ be a set of t databases, and let $Q_T(i, \sigma)$ denote the restriction of the k-tuple $\mathrm{query}_\mathcal{P}(i, \sigma)$ to databases in T. Now suppose that ϵ is an upper bound on the advantage of any circuit of size a in distinguishing between any two retrieval indices i_1, i_2 given the view of T in $\mathcal{P}$. It suffices to show that the same bound holds for the commodity scheme $\mathcal{C}_\mathcal{P}$ as well. Indeed, for any circuit C of size a we have:

$$|\mathbf{Pr}_{r,\sigma}[C(\langle Q_T(r,\sigma), i_1 - r\rangle) = 1] - \mathbf{Pr}_{r,\sigma}[C(\langle Q_T(r,\sigma), i_2 - r\rangle) = 1]| =$$

$$\left| \frac{1}{n} \sum_{r \in Z_n} \mathbf{Pr}_\sigma[C(\langle Q_T(r,\sigma), i_1 - r\rangle) = 1] - \frac{1}{n} \sum_{r \in Z_n} \mathbf{Pr}_\sigma[C(\langle Q_T(r,\sigma), i_2 - r\rangle) = 1] \right| =$$

$$\frac{1}{n} \left| \sum_{r \in Z_n} (\mathbf{Pr}_\sigma[C(\langle Q_T(r,\sigma), i_1 - r\rangle) = 1] - \mathbf{Pr}_\sigma[C(\langle Q_T(r + i_2 - i_1,\sigma), i_1 - r\rangle) = 1]) \right| \leq$$

$$\frac{1}{n} \sum_{r \in Z_n} |\mathbf{Pr}_\sigma[C(\langle Q_T(r,\sigma), i_1 - r\rangle) = 1] - \mathbf{Pr}_\sigma[C(\langle Q_T(r + i_2 - i_1,\sigma), i_1 - r\rangle) = 1]| \leq$$

$$\frac{1}{n} \cdot n\epsilon = \epsilon$$

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 $\mathcal{C}_\mathcal{P}$. $\square$

At this point it is important to observe that a collusion of the server with any single database can easily learn i, regardless of the privacy level 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 following sections.

6 Composing Commodity Schemes

In this section we show how commodity schemes can be composed to obtain schemes with stronger privacy properties.

6.1 The Single-Database Case

Consider any two single-database (computational) commodity schemes, $\mathcal{C}_1$ with servers $\mathcal{S}_1^1, \ldots, \mathcal{S}_{m_1}^1$, communication complexity (α_1, β_1) and commodity complexity $(\delta_1^u, \delta_1^{db})$, and $\mathcal{C}_2$ with servers $\mathcal{S}_1^2, \ldots, \mathcal{S}_{m_2}^2$, communication complexity (α_2, β_2) and commodity complexity $(\delta_2^u, \delta_2^{db})$. We define a composed commodity scheme, $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$, with a single database and $m = m_1 + m_2$ servers, as follows.

Commodities: Each of the servers $\mathcal{S}_1^1, \ldots, \mathcal{S}_{m_1}^1$ sends commodities as in $\mathcal{C}_1(K, n)$ and each of $\mathcal{S}_1^2, \ldots, \mathcal{S}_{m_2}^2$ sends commodities as in $\mathcal{C}_2(K, 2^{\alpha_1})$.

Retrieval:

1. The user computes a query q pointing to the original retrieval index i according to $\mathcal{C}_1(K, n)$ (with commodities given by servers of $\mathcal{C}_1$), and then a query q' pointing to the retrieval index q according to $\mathcal{C}_2(K, 2^{\alpha_1})$. The single query q' is sent to the database.
2. The database computes a virtual data string x', consisting of 2^{α_1} records of size β_1, where each record contains an answer to a corresponding user's query in $\mathcal{C}_1$. Specifically, the e-th record of x' consists of the answer, according to $\mathcal{C}_1$, to the retrieval query e, given the commodities distributed by $\mathcal{S}_1^2, \ldots, \mathcal{S}_{m_2}^2$. The database replies to the user's query by simulating $\mathcal{C}_2$ on x'. The user reconstructs x_i by first recovering $x'_{q'}$ from the answer, using the reconstruction function of $\mathcal{C}_2$, and then applying the reconstruction function of $\mathcal{C}_1$ to the resultant string.

The correctness of the composed scheme $\mathcal{C}$ easily follows from the correctness of $\mathcal{C}_1, \mathcal{C}_2$. We now analyze its complexity and level of privacy.

Complexity: Let $\pi_1 = (K, n)$ denote the parameters of the composed scheme (under which $\mathcal{C}_1$ is also used), and $\pi_2 = (K, 2^{\alpha_1(\pi_1)})$ denote the parameters under which $\mathcal{C}_2$ is used. The communication complexity of $\mathcal{C}$ is then $(\alpha_2(\pi_2), \beta_1(\pi_1) \cdot \beta_2(K, \pi_2))$, and its commodity complexity is $(max\{\delta_1^u(\pi_1), \delta_2^u(\pi_2)\}, max\{\delta_1^{db}(\pi_1), \delta_2^{db}(\pi_2)\})$. Assuming that block retrieval is implemented as discussed in Section 2 and that $\alpha_1 = \alpha_2 = \log n$, the communication complexity of $\mathcal{C}$ becomes $(\log n, \beta_1 \cdot \beta_2)$ and its commodity complexity $(max\{\delta_1^u, \delta_2^u\}, max\{\delta_1^{db}, \delta_2^{db}\})$. In particular, this means that a composition of arbitrarily many atomic schemes will yield a commodity scheme with query complexity $\alpha = \log n$. Finally, assuming $\alpha_1 = O(\log n)$, $\mathcal{C}$ will be computationally efficient if so are $\mathcal{C}_1, \mathcal{C}_2$.

Privacy (sketch): We claim that $\mathcal{C}$ keeps i (computationally) private from any collusion of parties S, such that *either* $\mathcal{C}_1$ keeps i private from $S_1 \stackrel{\text{def}}{=} S \cap \{\mathcal{DB}, \mathcal{S}_1^1, \ldots, \mathcal{S}_{m_1}^1\}$, *or* $\mathcal{C}_2$ keeps i private from $S_2 \stackrel{\text{def}}{=} S \cap \{\mathcal{DB}, \mathcal{S}_1^2, \ldots, \mathcal{S}_{m_2}^2\}$.

Let i_1, i_2 be any two retrieval indices. We denote by $view(i)$ the view of S when invoking $\mathcal{C}$ with retrieval index i. Similarly, for $b = 1, 2$, $view_b(i)$ denotes the view of S_b when invoking $\mathcal{C}_b$ with retrieval index i. Suppose first that $\mathcal{C}_1$ is private with respect to S_1. This means that $view_1(i_1) \approx_{\mathcal{F}} view_1(i_2)$ (where $\approx_{\mathcal{F}}$ denotes computational indistinguishability, parameterized by K, with respect to adversary class $\mathcal{F}$ – polynomial size circuits by default). Note that, since $\mathcal{C}_2$ is computationally efficient, $view(i)$ can be simulated in probabilistic polynomial time given a simulator for $view_1(i)$. Since $\mathcal{F}$ is polynomially closed, $view_1(i_1) \approx_{\mathcal{F}} view_1(i_2)$ implies that $view(i_1) \approx_{\mathcal{F}} view(i_2)$, as required. Now suppose that $\mathcal{C}_2$ is private with respect to S_2, hence $view_2(q_1) \approx_{\mathcal{F}} view_2(q_2)$ for any two intermediate queries q_1, q_2 picked by the user in the retrieval stage of $\mathcal{C}$. Since $view_1(i)$ and $view_2(q)$ are statistically independent for any i, q, it follows that $view(i_1) \approx_{\mathcal{F}} view(i_2)$.

Composed-Scheme-$\mathcal{C}_{\mathcal{P}}^{m}$ /* $\mathcal{P}$: 1-database computational PIR scheme; m: number of servers */

Commodities : Each server $\mathcal{S}_h$, $h = 1..m$, computes:

1. $r_h \leftarrow \$(Z_n)$;
   $\sigma_h \leftarrow \$(\mathrm{rand}_{\mathcal{P}})$
   $q_h \leftarrow \mathrm{query}_{\mathcal{P}}(r_h, \sigma_h)$;
   $z_h \leftarrow \text{reconstruct-info}_{\mathcal{P}}(r_h, \sigma_h)$;
2. $C_h^u \leftarrow (r_h, z_h)$;
   $C_h^{db} \leftarrow q_h$.

Retrieval :

1. $\mathcal{U} \rightarrow \mathcal{DB}$: $\Delta \stackrel{\text{def}}{=} i - \sum_{h=1}^{m} r_h \pmod{n}$.
2. $\mathcal{DB}$: $x^{(0)} \leftarrow x$;
   For $h \leftarrow 1$ to m, iteratively compute n-record data strings $x^{(h)}$, such that $x_e^{(h)} = \mathrm{answer}_{\mathcal{P}}(x^{(h-1)} << e, q_h)$, $e \in Z_n$.
   $\mathcal{DB} \rightarrow \mathcal{U}$: $a^{(m)} \stackrel{\text{def}}{=} x_{\Delta}^{(m)}$ /* only this entry of $x^{(m)}$ should be computed. */
3. $\mathcal{U}$: For $h \leftarrow m-1$ down-to 0, $a^{(h)} \leftarrow \mathrm{reconstruct}_{\mathcal{P}}(a_{h+1}, i - \sum_{e=1}^{h+1} r_e, z_{h+1})$;
   $\mathcal{U}$: Outputs $x_i = a^{(0)}$.

Figure 2: The single-database commodity scheme $\mathcal{C}_{\mathcal{P}}^{m}$

The above claim implies, in particular, that if $\mathcal{C}_1$ is s_1-private and $\mathcal{C}_2$ is s_2-private, then $\mathcal{C}$ is $(s_1 + s_2 + 1)$-private. A direct application of the composition tool to atomic schemes of the previous section thus gives the following.

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

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

Corollary 1. *For any constants $m \geq 1$ and $\varepsilon > 0$ there exists an m-server, single-database, $(m-1)$-private computational commodity scheme (assuming QRA), with communication complexity $(\log n, poly(K))$ and commodity complexity $(\log n, O(K \cdot n^{\varepsilon}))$.*

Proof. Such scheme can be obtained by applying Theorem 2 to the atomic commodity scheme $\mathcal{C}_{\mathcal{P}_1^{\varepsilon}}$. More precisely, the actual communication complexity is $(\log n, O(K^{m/\varepsilon}))$; for constant m and ε, this is polynomial in K.[4] □

[4] Fixing the number of databases [8, 1], or the complexity parameter ε [7, 15], has been the convention in other PIR related works. In the following section we will present a (multi-database) scheme whose communication complexity is also polynomial in m.

We note that since the scheme $\mathcal{P}_1^{\varepsilon}$ allows to trade answer complexity for query complexity (an extreme case is $\varepsilon = 1$, in which the query complexity is linear in n and the answer complexity is only K), a similar tradeoff can be established between *commodity* complexity and *communication* complexity in the m-server scheme of Corollary 1. Moreover, if a cheap and reliable source of public randomness is available, the scheme $\mathcal{P}_1^{\varepsilon}$ can be replaced by $\mathcal{P}_1^{\varepsilon\prime}$ to reduce the commodity cost asymptotically by a factor of K.

6.2 The Multi-Database Case

Aside from allowing information-theoretic security, multi-database PIR schemes possess the additional advantage of allowing the minimal answer complexity possible, namely a single answer bit per database (as in the schemes $\mathcal{P}_2^k, \mathcal{P}_3, \mathcal{P}_4^{t,d}$). This last feature is very useful in our context, since the communication bottleneck in composed schemes obtained via Theorem 2 is the answer complexity of their underlying PIR scheme. Another advantage of multi-database (information-theoretic) schemes is their superior performance on moderately large data strings, compared to their computational counterparts, both in terms of communication and computation.

However, when trying to apply the composition tool described in the previous subsection to multi-database commodity schemes, the following problem arises. Even in atomic multi-database schemes, the virtual data strings computed by different databases, as defined in the composition scheme, may differ; indeed, these strings depend on the different commodities given to the databases.. This puts us in position where there is no sufficient replication to allow using multi-database schemes for retrieving data from the virtual data strings.

The latter problem may be overcome by introducing enough additional replication to allow retrieval from the virtual data strings. This idea is used in the following multi-database generalization of Theorem 2.

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 exists an m-server, $(m-1, t)$-private, k^m-database commodity scheme $\mathcal{C}_{\mathcal{P}^k}^m$, with communication complexity $(\log n, \beta^m)$, commodity complexity $(\log n + \gamma, \alpha)$, and the same security type as $\mathcal{P}^k$.*

Proof. We generalize the composition tool of the previous subsection to the multi-database case. For any m_1-server, k_1-database commodity scheme $\mathcal{C}_1$ and m_2-server, k_2-database commodity scheme $\mathcal{C}_2$, define $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ as follows. $\mathcal{C}$ will use $m = m_1 + m_2$ servers $\{\mathcal{S}_{h_1}^1, \mathcal{S}_{h_2}^2 : h_1 \in [m_1], h_2 \in [m_2]\}$ as in the single-database case, and $k_1 k_2$ databases $\{\mathcal{DB}_{(j_1,j_2)} : j_1 \in [k_1], j_2 \in [k_2]\}$. Each server $\mathcal{S}_{h_1}^1$ sends commodities as in its original schemes, except that each commodity sent in the original scheme from $\mathcal{S}_h^1$ to $\mathcal{DB}_j$ is now sent to all databases $\mathcal{DB}_{(j,j_2)}, j_2 \in [k_2]$; similarly, each commodity originally sent from $\mathcal{S}_h^2$ to $\mathcal{DB}_j$ is now sent to all databases $\mathcal{DB}_{(j_1,j)}, j_1 \in [k_1]$. The retrieval stage proceeds as in the single-database case, except that the user reconstructs by first applying the reconstruction of $\mathcal{C}_2$ to each of the m_1 answer-tuples $(a_{j,1}, \ldots, a_{j,m_2})$, $j = 1, \ldots, m_1$, and then reconstruction of $\mathcal{C}_1$ to the m_1 reconstructed records. The arguments for correctness and privacy are very similar to those of the single-database case and are thus omitted. A scheme $\mathcal{C}_{\mathcal{P}^k}^m$ as required can be obtained by composing m atomic commodity schemes based on $\mathcal{P}^k$ in an arbitrary order. An explicit description of such composed scheme is given in Figure 3. $\square$

"Plugging in" the scheme $\mathcal{P}_3$, we obtain the following corollary.

Composed-Scheme-$\mathcal{C}^m_{\mathcal{P}^k}$ /* $\mathcal{P}^k$: k-database PIR scheme; m: number of servers */
The scheme uses k^m databases denoted $\mathcal{DB}_\tau$, $\tau = \tau_1..\tau_m \in [k]^m$.

Commodities : Each server $\mathcal{S}_h$, $h = 1..m$, computes:

1. $r_h \leftarrow \$(Z_n)$;
   $\sigma_h \leftarrow \$(\text{rand}_{\mathcal{P}})$
   $(q_h^1, \ldots, q_h^k) \leftarrow \text{query}_{\mathcal{P}^k}(r_h, \sigma_h)$;
   $z_h \leftarrow \text{reconstruct-info}_{\mathcal{P}}(r_h, \sigma_h)$;
2. $C_h^u \leftarrow (r_h, z_h)$;
   $C_h^{db_\tau} \leftarrow q_h^{\tau_h}$.

Retrieval :

1. $\mathcal{U} \rightarrow \mathcal{DB}_\tau$: $\Delta \stackrel{\text{def}}{=} i - \sum_{h=1}^m r_h \pmod n$, $\tau \in [k]^m$.
2. $\mathcal{DB}_\tau$: $x^\varepsilon \leftarrow x$; /* ε denotes the empty word */
   For $h \leftarrow 1$ to m, iteratively compute n-record data strings $x^{\tau'}$, such that $\tau' = \tau_1..\tau_h$, and $x_e^{\tau'} = \text{answer}_{\mathcal{P}^k}^{\tau_h}(x^{\tau_1..\tau_{h-1}} << e, q_h^{\tau_h})$, $e = 1..n$.
   $\mathcal{DB}_\tau \rightarrow \mathcal{U}$: $a_\tau \stackrel{\text{def}}{=} x_\Delta^\tau$ /* only this entry of x^τ should be computed. */
3. $\mathcal{U}$: For $h \leftarrow m-1$ down-to 0, for all $\tau' \in [k]^h$,
   $a_{\tau'} \leftarrow \text{reconstruct}_{\mathcal{P}}(a_{\tau'1}, a_{\tau'2}, \ldots, a_{\tau'k}, i - \sum_{e=1}^{h+1} r_e, z_{h+1})$;
   $\mathcal{U}$: Outputs $x_i = a_\varepsilon$.

Figure 3: The multi-database commodity scheme $\mathcal{C}^m_{\mathcal{P}^k}$

Corollary 2. *For any constant $m \geq 1$ there exists an m-server, $(m-1)$-private, 2^m-database computational commodity scheme, with communication complexity* $(\log n, 1)$ *and commodity complexity* $(\log n, K \cdot 2^{O(\sqrt{\log n})})$ *(assuming the existence of one-way functions).*

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 "practical" values of s such as 1 or 2.

7 Polynomial-Interpolation 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 $\log n, K$ 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 [8], based on the method of low-degree polynomial interpolation (cf. [5, 3, 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 will use the following two lemmas.

Lemma 1. *Let n be an integer and q a prime power, let $\vec{y}^h$ represent a sequence of variables $y_0^h, y_1^h, \ldots, y_{n-1}^h$, let e_i denote the i-th unit vector of length n (counting from $i = 0$), and let $\delta_{i,j}$ denote Kronecker's function (i.e., $\delta_{i,j}$ equals 1 if $i = j$ and 0 otherwise). Then for any $m \geq 1$ and $i \in Z_n$ there exists a degree-m multivariate polynomial $F_i^m(\vec{y}^1, \vec{y}^2, \ldots, \vec{y}^m)$ over $GF(q)$, such that for every $r_1, \ldots, r_m \in Z_n$,*

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

where $r = \sum_{h=1}^m r_h \bmod n$.
Moreover, F_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: $F_i^1(\vec{y}^1) = y_i^1$, and

$$F_i^m(\vec{y}^1, \ldots, \vec{y}^m) = \sum_{w \in Z_n} F_w^{m-1}(\vec{y}^1, \ldots, \vec{y}^{m-1}) \cdot y_{i-w}^m,$$

where the subtraction $i - w$ is taken modulo n. It easily follows by induction on h that F_i^m as defined above meets the specified requirements. Since $\vec{F}^h \stackrel{\text{def}}{=} (F_0^h, \ldots, F_{n-1}^h)$ can be efficiently evaluated given the values of $\vec{F}^{h-1}$, the values of $\vec{F}^m$ can be efficiently computed on a given assignment by iterating the evaluation of all $\vec{F}^h$, where h runs from 1 to m. $\square$

The next lemma slightly improves a similar bound implicit in [8].

Lemma 2. *Let l, d be positive integers, and $q > d + 1$ a prime power. Then there exist $n_{l,d} \stackrel{\text{def}}{=} \binom{l+d}{d}$ degree-d multivariate polynomials $p^i(y_1, \ldots, y_l)$, $0 \leq i < n_{l,d}$, and assignments $\vec{v}^i \in GF(q)^l$, $0 \leq i < n_{l,d}$, such that $p^{i_1}(\vec{v}^{i_2}) = \delta_{i_1,i_2}$ for all $0 \leq i_1, i_2 < n_{l,d}$.*

Proof. The existence of such $p^i, \vec{v}^i$ follows from the facts that: (1) the number of degree-d *monic monomials*[5] over $y_1, \ldots, y_l$ is $\binom{l+d}{d}$; and (2) when $d < q - 1$ these monomials are linearly independent, where each monomial p is identified in a natural way with the vector $\vec{u}^p \in GF(q)^{q^l}$ such that $u^p_{y_1 \ldots y_l} = p(y_1, \ldots, y_l)$. Indeed, since the $n_{l,d} \times q^l$ matrix whose rows are all the vectors $\vec{u}^p$ is of full rank, it is row-equivalent to a matrix A of which $n_{l,d}$ columns induce an identity matrix; identifying each such column with an assignment $\vec{v}^i$, 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 $p^i, \vec{v}^i$, slightly improving a construction from [8], is described in the following. Let $m^i(y_1, \ldots, y_l)$ be the i-th such monic monomial (say, according to lexicographic order). With each m^i associate a "characteristic vector" $\vec{v}^i = (v_1^i, \ldots, v_l^i)$, such that $m^i = \prod_{j=1}^l y_j^{v_j^i}$. Letting $y_0 \stackrel{\text{def}}{=} d - \sum_{j=1}^l y_j$ and $v_0^i \stackrel{\text{def}}{=} d - \sum_{j=1}^l v_j^i (= d - degree(m^i))$, define p^i as:

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

Since $\sum_{j=0}^l v_j^i = d$, each p^i is of degree d. It is straightforward to verify that the constructed $p^i, \vec{v}^i$ 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 [8], in a sense cannot be pushed any further.

[5] A degree-d monic monomial is a product of *at most* d, not necessarily distinct, variables.

Theorem 4. *Let m, t, d be positive integers, let $k \stackrel{def}{=} mtd+1$ 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 exists an $(m-1,t)$-private 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 $GF(q)$ (rather than single bits) at the same cost.*

Proof. Let $k = mtd+1$ and $l = l_{n,d}$, let $p^i, \vec{v}^i$ be as promised by Lemma 2, and $\vec{F}^m$ as promised by Lemma 1. We view the data bits (or records) as elements of $GF(q)$. 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 commodity to the user, and computes the corresponding assignment $\vec{v}^{r_h}$;
2. Independently shares each entry of $\vec{v}^{r_h}$ according to Shamir's secret sharing scheme with threshold $t+1$, over GF(q). Formally, for each w-th 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 $f_w^h(j)$, where f_w^h is a random degree-t (univariate) polynomial with free coefficient $v_w^{r_h}$, and j stands for the j-th nonzero element in $GF(q)$. We let $\vec{\mu}^{h,j}$ 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
$$a_j \stackrel{\text{def}}{=} \sum_{w \in Z_n} x_{w+\Delta} \cdot F_w^m(\vec{p}(\vec{\mu}^{1,j}), \ldots, \vec{p}(\vec{\mu}^{m,j})),$$
where $\vec{p} = (p^0, p^1, \ldots, p^{n-1})$, 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 $GF(q)$ such that $p(j) = a_j$, $j = 1, \ldots, k$.

PRIVACY: A collusion of $m-1$ servers and t databases learns nothing about the index r_h picked by the remaining server; indeed, since commodities sent from $\mathcal{S}_h$ to the databases are created according to Shamir's secret sharing scheme with threshold $t+1$, these commodities restricted to any t databases consist of uniformly and independently distributed field elements. It follows that the user's query reveals nothing about the retrieval index i.

CORRECTNESS: It suffices to show that the points (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 $(j, \mu_u^{h,j})$, $j \in [k]$, lie on a degree-t polynomial (namely, the polynomial f_u^h picked by the user) whose free coefficient is $v_u^{r_h}$. Since each p^w is of degree d, for any $h \in [m]$ and $w \in Z_n$ the points $(j, p^w(\vec{\mu}^{h,j}))$ lie on a degree-td polynomial whose free coefficient is $p^w(\vec{v}^{r_h})$, which by Lemma 2 equals δ_{e,r_h}. Finally, since each F_w^m is of degree d, for each $w \in Z_n$ the points $(j, F_w^m(\vec{p}(\vec{\mu}^{1,j}), \ldots, \vec{p}(\vec{\mu}^{m,j})))$ lie on a degree-mtd polynomial whose free coefficient is $F_w^m(e_{r_1}, \ldots, e_{r_m})$, which by Lemma 1 is equal to $\delta_{w,r}$ (where $r = \sum r_h$). It follows that the points (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. $\square$

REMARK: 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 exclusive-ors the answers to obtain x_i. We briefly describe how this is done. Observe that in $\mathcal{C}_{m,t,d}$ the user reconstruct x_i by computing a *fixed* linear combination over $GF(q)$ 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 $GF(q)$. 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 facts that $l_{n,d} = O(n^{1/d})$ for a constant d and $l_{n,d} = O(\log^2 n \log\log n)$ for $d = \frac{1}{3}\log n$ (see [8]), we have the following corollary of Theorem 4:

Corollary 3.

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

2. *For any constants s, t, there exists an (s,t)-private commodity scheme with $s+1$ servers, $k = \frac{1}{3}\log n \cdot t(s+1)+1$ databases, communication complexity $(\log n, 1)$ and commodity complexity $(\log n, O(\log^2 n \log\log n))$.*

8 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.[6]

We start with a motivating example. Suppose that the user wishes to retrieve two records, with (unrelated) indices i_1, i_2, using a 1-private single-database commodity scheme. A trivial solution to this problem is to independently invoke the scheme $\mathcal{C}^2_{\mathcal{P}}$, where $\mathcal{P}$ is some single-database computational PIR scheme, 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 4 (each server generates two pairs, one for each retrieval). Note that we 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 3 commodity pairs of the same size as before. Consider a scheme in which each of the 3 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

[6] A *q-query* scheme can be defined similarly to the original definition of a single-query scheme; the generalized privacy requirement requires that any two q-tuples of retrieval indices $(i_1, \ldots, i_q)$ and $(i'_1, \ldots, i'_q)$ will be indistinguishable (in the appropriate sense) by the databases.

commodity cost (3 commodities instead of 4). 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 $q \times m$ matrix over $GF(n)$ 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 2,3 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 $(\log n, \delta^{db})$. Then, there exists an m-server, k-database, $(d-1,t)$-private commodity scheme $\mathcal{C}_q^m$ for q retrievals, with communication complexity $q \cdot (\log n, \beta)$, and commodity complexity $(\log n, \delta^{db})$.*

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 an independent random element of $GF(n)$. We denote by $\mathcal{C}^w[c_1, \ldots, c_w]$, where $c_1, \ldots, c_w$ are fixed nonzero elements of $GF(n)$, a generalization of $\mathcal{C}^w$ in which the user's query is $i - \sum_{h=1}^{w} c_h r_h$ (i.e., $\mathcal{C}^w = \mathcal{C}^w[1,1,\ldots,1]$). In case of a scheme $\mathcal{C}^w$ obtained via composition of w atomic schemes (as in Theorems 2,3), such generalization can be realized by modifying the definition of the h-th composed atomic scheme so that the user's query is $\Delta = i - c_h r$ (instead of $i - r$), and each database replies with an answer on a database x' such that $x'_j = x_{c_h \cdot j + \Delta}$ (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 F_i^m from Lemma 1.

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

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

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

The correctness of $\mathcal{C}_q^m$ follows directly from the correctness of the schemes $\mathcal{C}^w$. 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 q retrievals in $\mathcal{C}^w$.

We now briefly sketch the proof of privacy. The joint distribution of queries sent by the user is $\vec{i} - G\vec{r}$, where $\vec{i} = (i_1, \ldots, i_q)$ are the retrieval indices, and $\vec{r} = (r_1, \ldots, r_m)$ is uniformly distributed in $GF(n)^m$. In addition, $m - d + 1$ of the commodities r_h are kept private from a collusion of t databases and $d-1$ servers. From the assumption that d is the minimal distance of the linear code generated by G it follows that for any restriction of $d-1$ of the r_h, the q-tuple $G\vec{r}$ is uniformly distributed over $GF(n)^q$, assuming that the unrestricted variables are uniformly and independently distributed over $GF(n)$ (see [20]). Thus, the view $\vec{i} - G\vec{r}$ keeps $\vec{i}$ private from any collusion of t databases and $d-1$ servers. $\square$

REMARK: The condition on G in Theorem 5 can be relaxed to allow rows with different Hamming weight in G; in such case, w can be taken as the *maximal* row weight in G. This generalization, however, will not be 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 multi-query scheme.

This restriction motivates the following problem in coding theory: Given a prime power n and positive integers q, d, find a minimal-length linear code over $GF(n)$ which is generated by codewords of weight d and whose minimal distance is d. We let $m(n,q,d)$ denote this minimal length, corresponding to the minimal number of commodity-tuples which by Theorem 5 are sufficient for performing q 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 $m(n,q,d)$ should be compared with the cost of the corresponding naive repetition scheme, whose d servers distribute a total of dq commodity-tuples. For instance, $m(n,q,2) = q+1$, as the $q \times (q+1)$ 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, q, d such that $n \geq q-1$, $m(n,q,d) = q+d-1$.*

Proof. For n, q, d as above, there exist $[q+d-1, q, d]$ linear codes over $GF(n)$ (see [16, Chapter 11]). The $q \times (q+d-1)$ generating matrix G of such code can be transformed via elementary row operations to a matrix G' generating the same code, which contains a $q \times q$ identity submatrix. Since the Hamming weight of each row of G' is at most $(q+d-1) - q + 1 = d$, we have shown that $m(n,q,d) \geq q+d-1$. On the other hand, it follows from Singleton bound that $m(n,q,d) \leq q+d-1$. □

We remark that the requirement $n \geq q-1$ is necessary for the above bound to hold. Luckily, in most plausible situations n is significantly larger than q,[7] in which case the following corollary of Fact 1 applies.

Corollary 4. *Assuming that the number of queries q is smaller than the database size n, Theorem 5 can asymptotically save a factor of $s+1$ in the amortized commodity cost of multi-query (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 $(s+1) \cdot \frac{m_0 - s}{m_0}$.*

9 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 composition of such schemes, are of this type. *Correctness* of such commodities is defined in the following way.

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

Definition 1. *Given a PIR scheme $\mathcal{P}$ and commodities $\mathcal{C}^u = (r, z)$, $\mathcal{C}^{db_j} = q_j$, $j = 1 \ldots k$, the commodities $(\mathcal{C}^u, \mathcal{C}^{db_1} \ldots \mathcal{C}^{db_k})$ are said to be* correct *on a data string x, if $reconstruct_{\mathcal{P}}(a_1 \ldots a_k, r, z) = x_r$, where $a_j = answer^j_{\mathcal{P}}(x, q_j)$; 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.*

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 commodities based on almost any PIR scheme 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 an honest behavior of the databases, none of them compromises the user's privacy, even when all databases and servers are dishonest. Finally, both procedures require a single round of (off-line) interaction, and their communication complexity involves an error probability parameter ϵ.

The following subsection describes the more efficient (and less general) testing procedure. The more general type is discussed in Appendix B.

9.1 Linear Schemes

Almost all known PIR schemes are *linear* in the following sense: reconstructing from the answers to any query-tuple $(q_1, \ldots, q_k)$, even from answers to badly formed queries, yields some linear combination of the data bits (or records) over a finite field $GF(q)$. When $q > 2$, as in polynomial interpolation based schemes, we view the records of x as elements of $GF(q)$. We remark that in order for the single-database scheme of [15] and its variants to satisfy this condition as well, the reconstruction information z must be consistent with the query; specifically, if the modulus N given as part of the query is equal to the the product of the two primes used for reconstruction, then the reconstructed bit will necessarily be equal to the exclusive-or of some subset of the data bits, depending on the query.

Our goal is thus to efficiently verify that the linear combination corresponding to the databases' commodities is correct, that is 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 $GF(q)^n$, 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 derived from constructions of small-bias probability spaces. The following fact has been proved by Naor and Naor [18].

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

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

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

1. The user picks a random index $a \in [|\mathcal{T}_{n,\epsilon}|]$ and sends it to each database.

2. Each database $\mathcal{DB}_j$ finds the a-th test-tuple in $|\mathcal{T}_{n,\epsilon}|$, $(w_1, \ldots, w_{l(n)})$, and replies with $(a_j^1, \ldots, a_j^{l(n)})$, where $a_j^b = \text{answer}_{\mathcal{P}}^j(w_b, q_j)$.
3. The user accepts if the commodities were correct on all $l(n)$ selected test strings; that is, if for every $1 \le b \le l(n)$, $\text{reconstruct}_{\mathcal{P}}(a_1^b \ldots a_k^b, r, z)$ is equal to the r-th entry of w_b.

For the single-database schemes based on quadratic residuosity, we add the following verification (which can be performed in parallel to the above procedure): the user sends the product $N_u = pq$ to the database, and the database confirms that N_u is equal to the modulus N supplied by the server.

The procedure clearly reveals nothing to the databases. 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 $v \ne e_r$ represents the linear combination corresponding to incorrect commodities, then $v - e_r \ne 0$, implying that with probability at least $1 - \epsilon$ there exists a test vector w_b from the selected test-tuple, such that $v \cdot w_b \ne e_r \cdot w_b$.

The communication complexity of the scheme is $O(\log n + \beta \log \frac{1}{\epsilon})$, 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 [18]) and amplify success probability by independent repetitions, yielding a computationally easier procedure with a slightly worse asymptotic communication of $O((\log n + \beta) \log \frac{1}{\epsilon})$.

Testing commodities of $\mathcal{C}_{m,t,d}$: Our definition of commodity correctness does not directly apply to the polynomial interpolation scheme from Section 7, as it is not composed of atomic schemes. However, it is possible to handle this scheme within the same linear framework as above. We briefly explain how this is done. In the scheme $\mathcal{C}_{m,t,d}$ (and also in its single answer-bit variant), each database $\mathcal{DB}_j$ interprets its commodity from S_h as an n-tuple of "shares" $s^{h,j} = (s_0^{h,j}, \ldots, s_{n-1}^{h,j})$. The commodities sent by $\mathcal{S}_h$ to the $k = mtd + 1$ databases are correct, if for every $i \in Z_n$, the points $(j, s_i^{h,j})$, $j = 1, \ldots, k$, lie on a degree-td polynomial over $GF(q)$ whose free coefficient is δ_{i,r_h}. To test that this is indeed the case, the same test set of data string as above can be used, where each $\mathcal{DB}_j$ replies on a test string x with $a_j = \sum_{i \in Z_n} x_i s_i^{h,j}$; then the user verifies that *all* answers a_j lie on the same degree-td polynomial whose free coefficient is x_{r_h}. Note that this final verification requires computing more than a *single* linear combination of the answers. However, if commodities from $\mathcal{S}_h$ are incorrect, then *some* inconsistency with the above requirement will be found with at least a $1 - \epsilon$ probability. This follows from observing that incorrectness of $\mathcal{S}_h$'s commodities implies the existence $td + 1$ databases that witness it; that is, the subset of the corresponding $td + 1$ share-vectors $s^{h,j}$ (uniquely) "interpolates" to something other than e_{r_h}. Hence, by reduction to the result about linear atomic schemes, we get that this inconsistency will be detected with at least a $1 - \epsilon$ probability.

10 Extensions

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

10.1 Protecting data privacy

An important advantage of known PIR schemes over the naive solution of "downloading" the entire data is their ability to control the amount of knowledge disclosed to the user. In [9] it is shown that if the databases are granted a source of shared randomness (which is hidden from the user), then known PIR schemes can be transformed into schemes which protect data privacy as well; that is, the transformed schemes prevent any user, even a dishonest one, from retrieving more than a single *physical* bit in each invocation. The communication overhead induced by these transformations is at most a multiplicative constant if the user is guaranteed to be honest, and $O(\log n)$ otherwise.

These results easily translate to the commodity setting. Moreover, assuming that the servers are honest, we may use the more efficient honest-user solutions (since the servers produce valid commodities), and also let the servers provide the required shared randomness between the databases. Finally, we note that the specific PIR schemes which we use to obtain low communication commodity schemes (see Section 3) have a very simple answer structure; this makes the protection of data privacy even easier (in fact, almost trivial) in most cases.

10.2 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 [19]. Private information storage schemes allow a user to privately write and read data to/from a data string which is *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 will only deal with *1-round* storage schemes; this is contrasted with the main scheme of [19], which requires logarithmically many rounds of interaction. We also assume that the "write" operation specifies an additive change to the i-th 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 i-th data bit). An "overwrite" operation can then be implemented using one read operation for retrieving the i-th record, followed by one write operation to change in its value as required.

Such t-private 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. In a "write" operation, the user sends to each database a *command*, which is interpreted by each database to represent some transformation of its share of the data string. The commands viewed by any t databases should give them no information (in the appropriate sense) on the write address or the change amount. 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.

The notion of *commodity storage scheme* can be defined in the natural way. We now argue that an analogue of the atomic commodity PIR schemes from Section 5 exists for storage schemes as well. Consider any 1-round k-database storage scheme in which the data string is shared record-wise (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. 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 two independent k-tuples of commands, one for flipping x_r and a "dummy" one for keeping x_r unchanged. The two k-tuples are sent as commodities to the databases in a random order, and the index r and a bit indicating the order of the two commands are sent to the user.
- In the on-line stage, the user sends to each database both a selection bit (depending on whether or not x_i should be flipped), and a shift amount $\Delta = i - r \pmod n$.

- Each database: (1) cyclically shifts its share by Δ places to the left; (2) interprets its selected commodity command and performs the required transformation on the shifted share; and (3) shifts the transformed share back by Δ places to the right.

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

Which concrete 1-round storage scheme can above transformation operate on? Such storage schemes can be based on any of the PIR schemes $\mathcal{P}_2^k$, $\mathcal{P}_3$ and $\mathcal{P}_4^{t,d}$, with similar storage cost as the retrieval cost of the PIR schemes.[8] In fact, such transformation can be applied to any *linear summation* PIR scheme, i.e. a scheme where the user's query is interpreted as asking for some linear combination of the data records, and reconstruction consists of computing some fixed linear combination (wlog, the sum) of the answers. 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. Implementing more general write operations (of changing x_i by λ) requires a slightly stronger assumption on the linear PIR schemes (namely, allowing the user to privately retrieve a selected multiple of x_i), which is met by all three schemes mentioned above.

What about commodity storage schemes with higher server-privacy threshold? The composition technique of Section 6 does not seem to to be applicable in its full generality to the case of storage. However, it is possible to directly construct *specific* multi-server storage schemes with similar parameters to the commodity PIR schemes of Corollary 2 or Theorem 4; to see this, it suffices to observe that in both of these schemes each database $\mathcal{DB}_j$ can combine the commodities from all servers to a single vector q_j, such that $\sum q_j = e_r$ (where r is the sum, modulo n, of commodity indices provided by the servers). In fact, this observation about Corollary 2 generalizes to any composition of *linear summation* schemes.

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

References

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

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

[3] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *STACS*, 1990.

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

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

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

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

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

[9] Y. Gertner, Y. Ishai, , E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. Manuscript, 1997.

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

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

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

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

[14] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of computer and systems sciences*, 28:270–299, 1984.

[15] E. Kushilevitz and R. Ostrovsky. Single-database computationally private information retrieval. In *Proc. of 38th FOCS*, 1997.

[16] F.J. Macwilliams and N.J. Sloane. *The theory of error correcting codes.* North Holland, 1978.

[17] E. Man. Personal communication, 1997.

[18] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proc. of 22th STOC*, pages 213–223, 1990.

[19] R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of 29th STOC*, pages 294–303, 1997.

[20] D.R. Stinson and J.L. Massey. An infinite class of counterexamples to a conjecture concerning non-linear resilient functions. *Journal of Cryptology*, 8:167–173, 1995.

[21] A.C. Yao. Protocols for secure computations (extended abstract). In *Proc. of 23rd FOCS*, pages 160–164, 1982.

A PIR Schemes with Low Answer Complexity

In this section we briefly describe the PIR schemes $\mathcal{P}_1^\varepsilon$, $\mathcal{P}_1^{\varepsilon\prime}$, $\mathcal{P}_2^k$, $\mathcal{P}_3$ and $\mathcal{P}_4^{t,d}$.

The scheme $\mathcal{P}_1^\varepsilon$ can be obtained by a straightforward modification of the recursive construction from [15]. A high-level description of a non-recursive version of $\mathcal{P}_1^\varepsilon$ is given in the following. The data string is viewed is a d-dimensional cube (where $d = 1/\varepsilon$ is a positive integer). The user encodes the d coordinates of the i-th position using quadratic characters modulo a randomly selected K-bit Blum integer $N = pq$. More specifically, each coordinate of the i-th position is encoded by a sequence of n^ε random residues modulo N with Jacobi symbol 1, all of which are *quadratic* except for the one corresponding to the value of that coordinate. The user sends all $d \cdot n^\varepsilon$ residues to the database, along with the modulus N. The database then performs d stages of "encoding", each of which takes a d'-dimensional data cube and returns an encoding (using quadratic characters) of the $(d'-1)$-dimensional cube obtained by restricting the d'-th coordinate to the corresponding value encoded by the user. (This "selection and encoding" mechanism uses the so-called XOR property of quadratic character). The output of this process (which encodes a 0-dimensional cube containing x_i alone) is sent back to the user. Finally, the user uses its trapdoor information (p, q) to perform a sequence of d "decoding" operations, reconstructing x_i. See [15] for more details.

The scheme $\mathcal{P}_1^{\varepsilon\prime}$ improves the query complexity of $\mathcal{P}_1^\varepsilon$ by an asymptotic factor of K (and in fact can improve the query complexity of the original scheme from [15] by the same factor), assuming that a public source of randomness is available. The idea is to modify the scheme $\mathcal{P}_1^\varepsilon$ from [15] by replacing each residue sent by the user with a public random residue. In order to control the quadratic characters of residues used by the database, the user will send a single "correction bit" for each public residue. Since (-1) is a quadratic non-residue with Jacobi symbol 1 modulo any Blum integer N, the database can flip the quadratic character of any public residue simply by negating it.

The scheme $\mathcal{P}_2^k$ was described in Section 3. The scheme $\mathcal{P}_3$ is obtained by a small modification of the 2-database computational scheme from [7]; under this modification, the user's queries are interpreted as two short pseudo-random "seeds", which are expanded (independently) by the two databases to two n-bit strings that XOR to e_i. The scheme can then proceeds as $\mathcal{P}_2^2$. Details of this scheme will appear in the journal version of [7]. Finally, the scheme $\mathcal{P}_4^{t,d}$ is obtained by modifying the polynomial-interpolation scheme from [8] according to the remark in Section 7.

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 a commodity or give statistical evidence that its 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 will be dispensed with in the sequel. For the sake of simplicity, we refer only to an *atomic* scheme, retrieving *bits* of data. The techniques can be adapted to any of our multi-server schemes as well.

Let L be a security parameter.

Commodity stage:

1. The server generates L independent commodity-tuples $C_1, \ldots, C_L$, each of which is distributed to the user and the databases.
2. The user and the databases parse the public random string as $y_1, y_2, \ldots, y_L$, where each y_a is n-bit long, and test each commodity on each data string. (The user, knowing the r-th bit of each test string, can determine whether the test succeeded).
3. If any of the commodities fails 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 $z_1, z_2, \ldots, z_L$, where each z_a is n-bit long.
2. The user and the databases invoke the original retrieval scheme L times, where in the a-th invocation the databases replace the data string x by the (random) string $x \oplus z_a$.
3. The user reconstructs L answer bits $b_1, \ldots, b_L$ according to the original reconstruction function, and outputs the majority vote of the bit values $b_a \oplus (z_a)_i$.

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

Claim 1. *The above testing procedure satisfies the following:*

- *If all commodities are correct, the user will always output the correct data bit.*
- *If the user does not reject at the commodity stage, then its output bit is wrong with probability* $2^{-\Omega(L)}$.

We now argue that when the servers are computationally bounded, public randomness can be replaced by a shorter seed sent from the user to each database.

Claim 2. *Suppose there exists a pseudo-random generator* $G : \{0,1\}^{\kappa(n)} \rightarrow \{0,1\}^n$. *Let* l *denote the 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 pseudorandom public 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.*

- *Assuming that the server is bounded to probabilistic-polynomial time (in n, L), the following holds: If the user does not reject at the commodity stage, then its output bit is wrong with negligible probability in n, L (where this time the probability space includes the coin-tosses of the server as well).*

Proof sketch. A server which fails to meet this requirement can be used in a straightforward way to distinguish between a random and a pseudo-random string. □

Substituting a "sufficiently secure" seed size for $\kappa(n)$ (e.g., $\kappa(n) = n^c$ for any $c > 0$ under standard security assumptions, or $\kappa(n) = polylog(n)$ under much more ambitious assumptions), we get a communication efficient testing procedure for the general case (though not quite as efficient as for the linear case).