General Constructions for Information-Theoretic Private Information Retrieval*

Amos Beimel† Yuval Ishai‡ Eyal Kushilevitz §

January 31, 2004

Abstract

A Private Information Retrieval (PIR) protocol enables a user to retrieve a data item from a database while hiding the identity of the item being retrieved; specifically, in a *t-private*, *k-server* PIR protocol the database is replicated among k servers, and the user's privacy is protected from any collusion of up to t servers. The main cost-measure of such protocols is the *communication complexity* of retrieving a single bit of data.

This work addresses the *information-theoretic* setting for PIR, where the user's privacy should be unconditionally protected against computationally unbounded servers. We present a general construction, whose abstract components can be instantiated to yield both old and new families of PIR protocols. A main ingredient in the new protocols is a generalization of a solution by Babai, Kimmel, and Lokam for a communication complexity problem in the multiparty simultaneous messages model.

Our protocols simplify and improve upon previous ones, and resolve some previous anomalies. In particular, we get: (1) 1-private k-server PIR protocols with $O(k^3 n^{1/(2k-1)})$ communication bits, where n is the database size; (2) t-private k-server protocols with $O(n^{1/\lfloor(2k-1)/t\rfloor})$ communication bits, for any constant integers $k > t \geq 1$; and (3) t-private k-server protocols in which the user sends $O(\log n)$ bits to each server and receives $O(n^{t/k+\epsilon})$ bits in return, for any constant integers $k > t \geq 1$ and constant $\epsilon > 0$. The latter protocols have applications to the construction of efficient families of *locally decodable codes* over large alphabets and to PIR protocols with reduced work by the servers.

Key words. private information retrieval, information-theoretic cryptography, locally decodable codes, multiparty communication complexity, simultaneous messages protocols.

*This paper contains the results of the two conference papers [22, 8].

†Dept. of Computer Science, Ben-Gurion University. E-mail: beimel@cs.bgu.ac.il. Homepage: www.cs.bgu.ac.il/∼beimel. Part of this research was done while the author was visiting DIMACS.

‡Department of Computer Science, Technion. E-mail: yuvali@cs.technion.ac.il. Research done in part while at DIMACS and AT&T Labs. Research supported in part by a grant from the Israel Science Foundation and by the Technion V.P.R. Fund.

§Department of Computer Science, Technion. E-mail: eyalk@cs.technion.ac.il. Homepage: www.cs.technion.ac.il/∼eyalk. Research supported in part by a grant from the Israel Science Foundation, by a grant from the Mitchell Schoref Fund and by the Technion V.P.R. Fund.

1 Introduction

A Private Information Retrieval (PIR) protocol allows a user to retrieve a data item of its choice from a database, so that the server storing the database does not gain information about the identity of the item being retrieved. For example, an investor might want to know the value of a specific stock without revealing which stock she is interested in. This problem was introduced by Chor, Goldreich, Kushilevitz, and Sudan [15], and has since then attracted a considerable amount of attention. In formalizing the problem, it is convenient to model the database by an n-bit string x, where the user, holding some *retrieval index* i, wishes to learn the i-th data bit x_i. This default setting can be easily extended to handle more general scenarios, e.g., of larger data items, several users, or several retrieved items per user.

A trivial solution to the PIR problem is to send the entire database x to the user. However, while providing perfect privacy, the *communication complexity* of this solution may be prohibitively large. Note that if the privacy constraint is lifted, then an optimal solution to the retrieval problem is to have the user explicitly send i to the server and receive x_i in return. This non-private solution requires only $\lceil \log_2 n \rceil + 1$ bits of communication, whereas the trivial private solution mentioned above requires n communication bits. Thus, the most significant goal of PIR-related research has been to minimize the communication overhead imposed by the privacy constraint.

Unfortunately, if the server is not allowed to gain *any* information about the identity of the retrieved bit, then the linear communication complexity of the trivial solution is optimal [15]. To overcome this problem, Chor et al. [15] suggested that the user accesses k replicated copies of the database kept on different servers, requiring that each *individual* server gets absolutely no information about i. PIR in this setting is referred to as *information-theoretic* PIR.[1] The above default privacy requirement naturally generalizes to t*-private* PIR, which keeps the index i private from any collusion of (at most) t out of the k servers.

The best PIR protocols known prior to the current work[2] are summarized below: (1) a 2-server protocol with $O(n^{1/3})$ communication bits [15]; (2) a k-server protocol with $O(n^{1/(2k-1)})$ communication bits, for any constant k (Ambainis [1] improving on [15]; see also Itoh [25]); (3) an $O(\log n)$-server protocol with $O(\log^2 n \log\log n)$ communication bits ([15], and implicitly in Beaver and Feigenbaum [5] and Beaver, Feigenbaum, Kilian, and Rogaway [6]), and (4) an $O(n^{1/\lfloor k/t \rfloor})$ bound for the more general case of t-private PIR [15]. All the above protocols require only a single round of interaction, in which the user sends a query to each server and receives an answer in return.

No strong general lower bounds for PIR are known. Mann [33] obtained a constant-factor improvement over the trivial $\log_2 n$ bound, for any constant k. In the 2-server case, much stronger lower bounds can be shown under various restrictions. Goldreich et al. [21] (generalizing a lower bound from [15]), obtain strong lower bounds for *linear* PIR protocols, where the user's query is interpreted as asking for some linear combination of the database entries, under the assumption that the user reads a constant number of bits from the answers it receives. Kerenidis and de Wolf [29] obtain nearly tight lower bounds for general (nonlinear) 2-server PIR protocols in which the servers' answers contain a constant number of bits (see also Beigel et al. [7]). Other lower bounds for restricted PIR protocols are given by Itoh [26]. Lower bounds for locally decodable codes, which are closely related to PIR (see Section 1.2), appear in [28, 17, 34, 29]. These results still leave an exponential gap between known upper bounds and lower bounds for unrestricted PIR.

A different approach for reducing the communication complexity of PIR is to settle for *computational* privacy, i.e., privacy against computationally bounded servers. Following a 2-server protocol by Chor and

[1] In principle, the term "information-theoretic PIR" may also refer to protocols which leak a negligible amount of information about i. However, there is still no evidence that such a relaxation is useful.

[2] Subsequent work will be addressed in Section 1.3.

Gilboa [14], Kushilevitz and Ostrovsky [31] showed that in this setting one server suffices. Under a standard number theoretic intractability assumption they construct, for every constant $\epsilon > 0$, a *single*-server protocol with communication complexity of $O(n^\epsilon)$ bits. Generalizations of this protocol were given in [36, 33]. Subsequently, Cachin, Micali, and Stadler [13] constructed, based on a new number theoretic intractability assumption, a single-server protocol with poly-logarithmic communication.[3]

From a practical point of view, single-server PIR protocols are preferable to multi-server ones for obvious reasons: they avoid the need to maintain replicated copies of the database or to compromise the user's privacy against several colluding servers. Moreover, single-server protocols from the literature obtain better asymptotic communication complexity than information-theoretic protocols with a constant number of servers. However, for real-life parameters the known single-server protocols are typically less efficient than known multi-server (even 2-server) protocols. Furthermore, single-server protocols have some *inherent* limitations which can only be avoided in a multi-server setting. For instance, it is impossible for a (sublinear-communication) single-server PIR protocol to have very short queries (say, $O(\log n)$ bits long) sent from the user to the server, or very short answers (say, one bit long) sent in return. These two extreme types of protocols, which can be realized in the information-theoretic setting, have various applications [18, 10]. Finally, the close relation between information-theoretic PIR and locally decodable codes [28], discussed in Section 1.2, further motivates the study of PIR in this setting.

1.1 Our Results

We present a unified general framework for the construction of PIR protocols, and instantiate its abstract components to give three families of protocols.

Main family. Protocols from this family are based on a generalization of a multiparty communication protocol by Babai, Kimmel, and Lokam [4] in the so-called simultaneous messages model. They provide the following improvements to the previous state of the art.

- 1-private k-server PIR with $O(k^3 n^{1/(2k-1)})$ communication bits (compared to $O(2^{k^2} n^{1/(2k-1)})$ in [1], and $O(k!\, n^{1/(2k-1)})$ in [25]). The polynomial dependence on k allows to get PIR with *polylogarithmic* communication, where for $k = \Omega(\log n/ \log\log n)$ the improvement over previous polylog-communication protocols from [15, 6] is nearly quadratic. Similar improvements are obtained for the related "instance hiding" problem [5, 6]. We note that even in the 2-server case, the communication complexity of the protocol from [15] is improved by a constant factor.
- t-private k-server PIR with communication complexity $O(n^{1/\lfloor(2k-1)/t\rfloor})$, for every constant k. For $t > 1$, this is a nearly quadratic improvement over the previous state of the art [15].
- Efficient PIR with logarithmic query length. Specifically, our main family can be used to obtain t-private k-server PIR protocols with $O(\log n)$ query bits and $O(n^{t/k+\epsilon})$ answer bits, for every constant integers k, t such that $1 \le t < k$ and constant $\epsilon > 0$. Protocols of this type (with $t = 1$) were used in [10] to save computation in PIR via preprocessing, and have interesting applications, discussed in Section 1.2 below, to the construction of efficient locally decodable codes over large alphabets.

Binary family. Protocols from this family are based on Shamir's secret sharing scheme and polynomial interpolation. This family generalizes protocols from [5, 6, 15, 18], and allows t-private k-server PIR

[3] A different single-server protocol with poly-logarithmic communication complexity was suggested in [30]. However, the underlying intractability assumption has been recently broken [16, 12].

where the user sends to each server a query of length $O(\frac{k}{t}\log kn^{1/\lfloor(k-1)/t\rfloor})$ and receives a single bit in return (hence the term *binary*). Protocols from this family are useful for obtaining: (1) binary locally decodable codes (see Section 1.2); (2) efficient PIR protocols for retrieving large records [15] or "streams" of data; (3) PIR protocols with an optimal amount of *on-line* communication (see [18]); and (4) PIR protocols with a poly-logarithmic amount of *on-line* work by the servers (see [10]).

Probe-efficient family. The *probe complexity* of a PIR protocol measures the number of bits from each answer which are actually read by the user. Our third family of protocols utilizes a generalization of the "combinatorial cubes" geometry, first used in [15], to obtain probe-efficient PIR. Protocols from this family have a very low (typically constant) probe complexity and a better communication complexity than the protocols from the binary family. In contrast, protocols from the main family (for $t > 1$) require the user to read all of the bits returned by the servers, but have a better communication complexity.[4] The probe complexity of PIR is an interesting complexity measure for several reasons. First, probe-efficient protocols enable the user to be very efficient in space: In all of our probe-efficient protocols the user's algorithm requires only $O(\log n)$ space, whereas other protocols require the user to remember either the queries sent to the servers or the randomness used to generate them. Second, probe-efficient PIR protocols allow the user to make use of another PIR protocol for retrieving only the bits that it needs to read.[5] This feature may be useful for obtaining recursive information-theoretic PIR protocols [1, 9] and was used for obtaining time-efficient computational PIR protocols via preprocessing [10]. Third, a low probe complexity can be significant in the context of the coding application discussed in Section 1.2 below. Finally, the lower bounds proved in [21] are also phrased in terms of (and depend on) the probe complexity.

Our main upper bounds on the complexity of PIR are summarized in Table 1.

<table>
<tr><th>Family</th><th>t</th><th>Query length</th><th>Answer length</th><th>Probes</th><th>Reference</th></tr>
<tr><td rowspan="4">Main</td><td>1</td><td>$O(k^2n^{1/(2k-1)})$</td><td>$O(kn^{1/(2k-1)})$</td><td>$2k-1$</td><td>Cor. 6.6</td></tr>
<tr><td>any</td><td colspan="3">$O(\frac{k}{t}\binom{k}{t}n^{1/\lfloor(2k-1)/t\rfloor})$</td><td>Cor. 6.2</td></tr>
<tr><td>1</td><td>$O_{k,\epsilon}(\log n)$</td><td>$O_{k,\epsilon}(n^{1/k+\epsilon})$</td><td>$O_{k,\epsilon}(n^{\epsilon})$</td><td>Cor. 6.6</td></tr>
<tr><td>any</td><td>$O_{k,\epsilon}(\log n)$</td><td colspan="2">$O_{k,\epsilon}(n^{t/k+\epsilon})$</td><td>Cor. 6.3</td></tr>
<tr><td>Binary</td><td>any</td><td>$O(\frac{k}{t}\log kn^{1/\lfloor(k-1)/t\rfloor})$</td><td colspan="2">1</td><td>Cor. 6.10</td></tr>
<tr><td rowspan="3">Probe-efficient</td><td>2</td><td colspan="2">$O_k(n^{1/\lfloor(2k+1)/3\rfloor})$</td><td>$\lfloor(2k+1)/3\rfloor$</td><td>Cor. 6.13</td></tr>
<tr><td>3</td><td colspan="2">$O_k(n^{1/\lfloor 2k/5\rfloor})$</td><td>$\lfloor 2k/5\rfloor$</td><td>Cor. 6.13</td></tr>
<tr><td>$\geq\sqrt{k}$</td><td colspan="2">$O_k(n^{1/(\lfloor k/t\rfloor+1)})$</td><td>$\lfloor k/t\rfloor+1$</td><td>Cor. 6.13</td></tr>
</table>

Table 1: Summary of the main upper bounds in this paper. The *query length* is the maximal number of bits sent from the user to any single server, and the *answer length* is the maximal number of bits sent in return. The number *probes* is the number of bits the user needs to read in each answer. The notation $O_{y_1,..,y_s}(\cdot)$ indicates that $y_1,\ldots,y_s$ are treated as constants.

[4] In the case of 1-private PIR, protocols from the main family can typically achieve the same probe complexity and a slightly better communication complexity.

[5] Note that in order to effectively apply this idea in an information-theoretic setting, one must ensure that the answers in the "external" protocol contain sufficient replication.

1.2 Locally Decodable Codes

Much of the recent interest in information-theoretic PIR is due to its close relation with *locally decodable codes* (LDC). Standard error-correcting codes can provide high fault tolerance while only moderately expanding the encoded message. However, their decoding procedure requires to read the entire encoded message even if one is only interested in decoding a single bit of this message. LDC simultaneously provide high fault tolerance and a sublinear-time "local" decoding procedure. To make this possible, the decoding procedure must use randomness for selecting which bits to probe, and some error probability must be tolerated. More formally, a code $C : \{0,1\}^n \to \Sigma^m$ is said to be (k, δ, ρ)-*locally decodable* if every bit x_i of x can be decoded from $y = C(x)$ with success probability$\geq 1/2+\rho$ by reading k (randomly chosen) symbols of y, even if up to a δ-fraction of the symbols in y were adversarially corrupted. By default, a k-*query LDC* over Σ is a family of $(k, \delta(n), \rho(n))$-locally decodable codes $C_n : \{0,1\}^n \to \Sigma^{m(n)}$ such that $\delta(n), \rho(n)$ are lower bounded by some positive constant, independent of n.

Katz and Trevisan [28] have shown an intimate connection between LDC and information-theoretic PIR. In particular, any 1-private k-server PIR protocol with query length $\alpha(n)$ and answer length $\beta(n)$ can be used to construct a k-query LDC of length $O(2^{\alpha(n)})$ over $\Sigma = \{0,1\}^{\beta(n)}$. (If the query to each server is uniformly distributed over its domain, as is the case for all of the protocols in this paper, the encoding of a string $x \in \{0,1\}^n$ can be simply defined to be the concatenation of the servers' answers to all possible queries.) This motivates the construction of PIR protocols with short queries.[6]

Our constructions of PIR protocols with logarithmic query length have an interesting interpretation in terms of LDC. The main focus of the recent work on LDC [28, 21, 17, 34, 29] has been on finding the minimal asymptotic length $m(n)$ of a k-query LDC over a *binary* alphabet, or a constant size alphabet. Generalizing a PIR lower bound of [33], it is proved in [28] that for any constant k the code length must be super-linear. In the case of 2-query LDC, exponential lower bounds were obtained in [21] (for linear codes) and [29] (for general codes). While no super-polynomial lower bounds are known for $k > 2$, the best known upper bounds (obtained from PIR protocols with a single answer bit per server) are exponential in n. This work has relevance to the following dual question: Suppose that we insist on the code being *efficient*, namely of polynomial length, but allow the alphabet Σ to grow with n. Then, how small can $\Sigma(n)$ be? Our results imply the following nontrivial upper bound: for any constant $\epsilon > 0$ it suffices to let $\Sigma(n) = \{0,1\}^{\beta(n)}$, where $\beta(n) = O(n^{1/k+\epsilon})$.

1.3 Subsequent Work

In a recent work, Beimel, Ishai, Kushilevitz, and Raymond [9] construct, for any fixed k, a family of 1-private k-server PIR protocols with $O(n^{1/c_k})$ communication bits where $c_k = \Omega(k \log k / \log\log k)$. This is a significant asymptotic improvement over the best 1-private protocols presented in this work, for which $c_k = 2k - 1$. The construction of [9] uses a recursive composition of probe-efficient protocols, which are based on (and in a sense generalize) ones appearing in this work. We would like to stress though that the aforementioned bound of [9] does not subsume the results of this work. In particular, it only applies to 1-private PIR (a generalization to $t > 1$ seems nontrivial) and its big-O notation hides a super-exponential dependence on k (in contrast to a polynomial dependence on k in our 1-private protocols). Consequently, the protocols presented in this work still provide the best known upper bounds on PIR in many cases, including t-private PIR for $t > 1$, PIR with short queries, or PIR with polylogarithmic communication.

[6] A relaxed notion of information-theoretic PIR (allowing some limited information leakage of i and some error probability) is sufficient for obtaining a corresponding LDC. However, to date there is no evidence that this type of relaxation can significantly help, as all known constructions of LDC directly correspond to known (perfect) PIR protocols.

ORGANIZATION. In Section 2 we give an overview of our unified approach for constructing PIR protocols. In Section 3 we provide some necessary definitions. In Section 4 we describe a meta-construction of PIR protocols, in Section 5 we instantiate one of its crucial ingredients, and in Section 6 we derive new and old families of PIR protocols as instances of the meta-construction. Finally, in Section 7 we obtain an optimization of previous protocols.

2 Overview of Techniques

At the heart of our constructions is a novel combination of two techniques.[7]

2.1 Reduction to Polynomial Evaluation

The first technique is a reduction of the retrieval problem to the problem of multivariate polynomial evaluation. Specifically, the retrieval of x_i, where the servers hold x and the user holds i, is reduced to an evaluation of a multivariate polynomial p_x, held by the servers, on a point $E(i)$, which the user determines based on i. We refer to $E(i)$ as the *encoding* of i. As observed in [6] and, more generally, in [15], the degree of p_x can be decreased by increasing the length of the encoding $E(i)$ (i.e., the number of variables in p_x). Originating in [5], different variants of this reduction have been implicitly or explicitly used in virtually every PIR-related construction. In fact, even the seemingly "combinatorial" constructions from [15, 1] can be cast in this terminology (see Section 6.3). It is interesting to note that encodings realizing the optimal length-degree tradeoff, which were utilized in [15, 18] to construct special families of PIR protocols with short answer length, could not be used to realize the best known bounds on the *total* communication complexity. In [15, 1] it seemed necessary to use a more redundant encoding for obtaining the best protocols. This situation is remedied in the current work, where the most communication-efficient protocols are obtained using an optimal encoding. Consequently, we get at least a constant-factor improvement to the communication complexity of all previous protocols, including in the 2-server case.

2.2 Simultaneous Messages Protocols for Polynomial Evaluation

A main ingredient in our new protocols is a generalization of a solution by Babai, Kimmel, and Lokam [4] to a communication complexity problem of computing the *generalized addressing function* in the so-called *simultaneous messages* (SM) model. Interestingly, this problem was motivated by circuit lower bounds questions, completely unrelated to privacy or coding. Toward solving their problem, Babai et al. consider the following scenario. A degree-d m-variate polynomial p is known to k players, and k points $y_1, y_2, \ldots, y_k$ (each being an m-tuple of field elements) are distributed among the players such that player j knows all points except y_j. An external referee knows *all* k points y_j but does not know p. How efficiently can the value $p(y_1 + y_2 + \ldots + y_k)$ be communicated to the referee if the players are restricted to simultaneously sending a single message (each) to the referee?

A naive solution to the above problem is to have one of the players send an entire description of p to the referee. Knowing all points y_j, the referee can then easily compute the required output. A key observation made in [4] is that it is in fact possible to do much better. By decomposing $p(y_1 + y_2 + \ldots + y_k)$ into monomials and assigning each monomial to a player having the least number of unknown values, it is possible to write $p(y_1 + \ldots + y_k)$ as the sum of k *lower degree* polynomials in the inputs, each known to one of the players. More precisely, each player j can locally compute from its inputs a degree-$\lfloor d/k \rfloor$ polynomial

[7] A restricted use of this approach was made in [10].

p_j with its *unknown* inputs y_j as indeterminates, such that $p(y_1+\ldots+y_k) = p_1(y_1)+p_2(y_2)+\ldots+p_k(y_k)$. Then, by letting each player j communicate the (much shorter) description of p_j, the referee can compute the required output. The amount of savings obtained by this degree reduction technique depends on the values of the parameters m,d, and k. In [4, 2], due to constraints imposed by the specific problem they consider, the degree-reduction technique is applied with rather inconvenient choices of parameters. Thus, in their setting the full savings potential of the technique has not been realized. It turns out that in the PIR context, where there is more freedom in the choice of parameters, the full spectrum of possible tradeoffs is revealed.

It is instructive to look at three useful choices of parameters: (1) If $d = 2k-1$, then the degree of each polynomial p_j is only $\lfloor (2k-1)/k \rfloor = 1$. When $m \gg d$, this $2k-1$ savings factor in the degree makes the description size of each p_j roughly the $(2k-1)$-th root of the description size of p. (2) If $d = k-1$, the degree of each p_j becomes 0, and consequently communicating each p_j requires sending only a single field element. (3) Finally, if $m \gg d$ and $d \gg k$, then the cost of communicating p_j is roughly the k-th root of the cost of communicating p. These three examples, respectively, turn out to imply the existence of k-server PIR protocols with: (1) both queries and answers of length $O(n^{1/(2k-1)})$; (2) queries of length $O(n^{1/(k-1)})$ and answers of length $O(1)$; (3) queries of length $O(\log n)$ and answers of length $O(n^{1/k+\epsilon})$, for an arbitrarily small constant $\epsilon > 0$.

2.3 Combining the Two Techniques

In the case of 1-private PIR, the two techniques can be combined in the following natural way. On input i, the user computes an encoding $y = E(i)$ and the servers compute a degree-d polynomial p_x such that $x_i = p_x(E(i))$. To generate its queries, the user "secret-shares" $E(i)$ among the servers by first breaking it into otherwise-random vectors $y_1, \ldots, y_k$ which add up to y, and then sending to each server $\mathcal{S}_j$ all vectors except y_j. Using the SM communication protocol described in the previous section, the servers communicate $x_i = p_x(y)$ to the user.

This simple combination of the two techniques is already sufficient to yield some of the improved constructions. In the remainder of this work we generalize and improve the above solution in several different ways. First, we abstract its crucial components and formulate a generic "meta-construction" in these abstract terms. Second, we instantiate the abstract components to accommodate more general scenarios, such as the one required for dealing with t-private PIR. Third, we try to optimize the *probe complexity* of the above SM protocol, allowing the user to recover x_i by reading just a few bits from each answer. Finally, for both the 1-private and the t-private cases, we attempt at optimizing the amount of replication in the setting of [4] while maintaining the quality of the solution (that is, we use a more efficient secret-sharing scheme for distributing $E(i)$). These generalizations motivate various extensions of the SM communication model, which may be of independent interest.

3 Definitions

Notation. By $[k]$ we denote the set $\{1, \ldots, k\}$, and by $\binom{[k]}{t}$ all subsets of $[k]$ of size t. For a k-tuple v and a set $T \subseteq [k]$, let v_T denote the restriction of v to its T-entries. That is, if $T = \{i_1, \ldots, i_t\}$ and $v = (v_1, \ldots, v_k)$ then $v_T = (v_{i_1}, \ldots, v_{i_t})$. By Y_j for some j we represent a variable, while by the lower-case letter y_j we represent an assignment to the variable Y_j. By H we denote the binary entropy function; that is, $H(p) = -p\log p - (1-p)\log(1-p)$, where in this paper all logarithms are taken to the base 2.

Polynomials. Let GF(q) denote the finite field of q elements. By $F[Y_1, \ldots, Y_m]$ we denote the linear space of all polynomials in the indeterminates $Y_1, \ldots, Y_m$ over a field F, and by $F_d[Y_1, \ldots, Y_m]$ its subspace consisting of all polynomials whose *total* degree is *at most* d, and whose degree in each indeterminate is at most $|F|-1$. A natural basis for this linear space consists of all *monic monomials* satisfying the above degree restriction. Restricting the degree to be at most $|F|-1$ guarantees that each polynomial in $F_d[Y_1, \ldots, Y_m]$ represents a distinct function from F^m to F. It follows that the "syntactic dimension" of $F_d[Y_1, \ldots, Y_m]$, which is equal to the number of distinct monic monomials, coincides with its "semantic dimension", the dimension of the subspace of $F^{|F|^m}$ spanned by $\{(p(y))_{y \in F^m} : p \in F_d[Y_1, \ldots, Y_m]\}$.

The case $F = \text{GF}(2)$ will be the most useful in this work. In this case, the natural basis consists of all products of at most d distinct indeterminates. Hence, $\dim(F_d[Y_1, \ldots, Y_m]) = \sum_{w=0}^{d} \binom{m}{w}$ for $F = \text{GF}(2)$. We denote this dimension by $\Lambda(m, d) \stackrel{\text{def}}{=} \sum_{w=0}^{d} \binom{m}{w}$. We will also be interested in $F_d[Y_1, \ldots, Y_m]$ where $|F| > d$. In this case, the dimension of the space is $\binom{m+d}{d}$.

3.1 PIR Protocols

In this section we define 1-round information-theoretic PIR protocols.[8] A k-server PIR protocol involves k servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$, each holding the same n-bit string x (the database), and a user $\mathcal{U}$ who wants to retrieve a bit x_i of the database.

Definition 3.1 (PIR) *A k-server PIR protocol $\mathcal{P} = (\mathcal{R}, \mathcal{Q}_1, \ldots, \mathcal{Q}_k, \mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{C})$ consists of a probability distribution $\mathcal{R}$ and three types of algorithms: query algorithms $\mathcal{Q}_j(\cdot, \cdot)$, answering algorithms $\mathcal{A}_j(\cdot, \cdot)$, and a reconstruction algorithm $\mathcal{C}$. At the beginning of the protocol, the user picks a random string r from the distribution $\mathcal{R}$. For $j = 1, \ldots, k$, it computes a query $q_j = \mathcal{Q}_j(i, r)$ and sends it to server $\mathcal{S}_j$. Each server $\mathcal{S}_j$ responds with an answer $a_j = \mathcal{A}_j(q_j, x)$ (we assume, without loss of generality, that the servers are deterministic; otherwise, we can fix their randomness arbitrarily. Hence, the answer is a* function *of the query and the database). Finally, the user computes the bit x_i by applying the reconstruction algorithm $\mathcal{C}(i, r, a_1, \ldots, a_k)$. A k-server protocol as above is a t*-private *PIR protocol, if it satisfies the following requirements:*

Correctness. *The user always computes the correct value of x_i. Formally, for every $i \in [n]$, every random string r, and every database $x \in \{0,1\}^n$,*

$$\mathcal{C}(i, r, \mathcal{A}_1(\mathcal{Q}_1(i, r), x), \ldots, \mathcal{A}_k(\mathcal{Q}_k(i, r), x)) = x_i.$$

t-Privacy. *Each collusion of up to t servers has no information about the bit that the user tries to retrieve. Formally, for every two indices $i_1, i_2 \in [n]$ and for every $T \subseteq [k]$, of size $|T| \leq t$, the distributions $\mathcal{Q}_T(i_1, \cdot)$ and $\mathcal{Q}_T(i_2, \cdot)$ are identical.*

The *communication complexity* of a PIR protocol is the total number of bits communicated between the user and the k servers, maximized over all choices of x, i and r. We will often separately measure the *query length* and the *answer length*. We say that a PIR protocol has query length α and answer length β if the user sends at most α bits to *each server*, and receives at most β bits from each server. Note that in such a case, the communication complexity is at most $k(\alpha + \beta)$.

[8] All the protocols constructed in this paper, as well as all previous information-theoretic PIR protocols, require a single round of queries and answers. This definition may be extended to multi-round PIR in the natural way.

While the main complexity measure of PIR protocols is their communication complexity, we will sometimes also be interested in their *probe complexity*, which measures the number of bits that the user actually has to read from the servers' answers. We say that the probe complexity of a PIR protocol is c if the user can compute its output by probing at most c bits from each answer (whose location may depend on its input i and the choice of r).

3.2 Linear Secret-Sharing

A *t-private secret-sharing scheme* allows a dealer to distribute a secret s among k players, such that any set of at most t players learns nothing about s from their joint shares, and any set of at least $t+1$ players can completely recover s from their shares. A secret-sharing scheme is said to be *linear over a field F* if $s \in F$, and the share received by each player consists of one or more linear combinations of the secret and u independently random field-elements (where the same random field-elements are used for generating all shares). A linear secret-sharing scheme over a field f is formally defined by a k-tuple $L = (L_1, \ldots, L_k)$ such that each L_j is a linear mapping from $F \times F^u$ to F^{ℓ_j}, where ℓ_j is the j-th player share length. To share a secret $s \in F$ the dealer chooses a random $r \in F^u$ with uniform distribution and the share of the jth player is $L_j(s, r)$ (the same r is used for all the parties). Finally, given a linear secret-sharing scheme as above, a vector in F^m will be shared by independently sharing each of its m entries. We next define two examples of linear secret-sharing schemes that will be useful in the paper.

Definition 3.2 (Shamir's scheme [35]) Let $F = \mathrm{GF}(q)$, where $q > k$, and let $\omega_1, \ldots, \omega_k$ be distinct nonzero elements of F. To t-privately share a secret $s \in F$, the dealer chooses t random elements $a_1, \ldots, a_t$, which together with the secret s define a univariate polynomial $p(Y) \stackrel{\text{def}}{=} a_t Y^t + a_{t-1} Y^{t-1} + \ldots + a_1 Y + s$. Observe that $p(0) = s$. The share of the j-th player is $p(\omega_j)$. This share is a linear combination of the random inputs and the secret. Each set of at least $t+1$ players can recover $p(Y)$ by interpolation, and hence can also reconstruct $s = p(0)$. On the other hand, every set of t players learns nothing about s from their shares.

Definition 3.3 (The CNF scheme [24]) This scheme may be defined over any finite field (in fact, over any finite group), and proceeds as follows. To t-privately share a secret $s \in F$:

- *Additively* share s into $\binom{k}{t}$ pieces, each labeled by a different set from $\binom{[k]}{t}$; that is, $s = \sum_{T \in \binom{[k]}{t}} r_T$, where the pieces r_T are otherwise-random field elements. (Equivalently, all r_T except one may be chosen uniformly at random, and the last is determined so that they all sum up to s.)
- Distribute to each player P_j all pieces r_T such that $j \notin T$.

The t-privacy of the above scheme follows from the fact that every t players miss exactly one piece r_T (namely, the one labeled by their index set). Every set of $t+1$ players views all pieces, and can therefore reconstruct the secret. The share of each party consists of $\binom{k-1}{t}$ field elements.[9]

4 The Meta-Construction

In this section we describe our construction in terms of its abstract general components, and specify some useful instantiations for each of these components. In Section 5 several combinations of these instantiations are used for obtaining different families of PIR protocols.

[9] This scheme has also been referred to as *replication-based* secret-sharing [20]. It may be viewed as a special case of the formula-based secret-sharing construction from [11], obtained by using the canonical CNF representation of the threshold function.

4.1 Building Blocks

There are three parameters common to all of our constructions: (1) a finite *field* F, (2) a *degree* parameter d, and (3) an *encoding length* parameter m. The database x is always viewed as a vector of n field elements.[10] Some variants of our construction use an additional *segment length* parameter ℓ.

All variants of our construction (as well as previous PIR protocols) can be cast in terms of the following abstract building blocks:

Linear space of polynomials. Let $V \subseteq F_d[Y_1, \ldots, Y_m]$ be a linear space of degree-d m-variate polynomials such that $\dim(V) \geq n$. The three most useful special cases are:

V1: The space $F_d[Y_1, \ldots, Y_m]$ where $F = \text{GF}(2)$; in this case, m and d must satisfy $\Lambda(m, d) \geq n$. We will use the notation $\mathbf{V1}_{d,m}$ whenever we want to emphasize the underlying parameters d and m.

V2: The space $F_d[Y_1, \ldots, Y_m]$ where $|F| > d$; in this case, m and d must satisfy $\binom{m+d}{d} \geq n$. Again, we use $\mathbf{V2}_{d,m}$ to emphasize the parameters d and m.

V3: The linear subspace of $F_d[Y_1, \ldots, Y_m]$ such that $F = \text{GF}(2)$ and V is spanned by the following basis of monomials. Let ℓ be an additional *segment length* parameter, and let $m = \ell d$. We label the m indeterminate by $Y_{g,h}$, where $g \in [d]$ and $h \in [\ell]$. The basis of V will include all monic monomials containing exactly one indeterminate from each segment, i.e., all monomials of the form $Y_{1,h_1} Y_{2,h_2} \cdots Y_{d,h_d}$. Since the number of such monomials is ℓ^d, the restriction on the parameters in this case is $\ell^d \geq n$.

Low-degree encoding. A *low-degree encoding* (with respect to the polynomial space V) is a mapping $E : [n] \to F^m$ satisfying the following requirement: There exist m-variate polynomials $p_1, p_2, \ldots, p_n \in V$ such that $\forall i, j \in [n]$, $p_i(E(j))$ is 1 if $i = j$ and is 0 otherwise. By elementary linear algebra, $\dim(V) \geq n$ is a necessary and sufficient condition for the existence of such an encoding. Given a low-degree encoding E and polynomials $p_1, p_2, \ldots, p_n$ as above, we will associate with each database $x \in F^n$ the polynomial $p_x \in V$ defined by $p_x(Y_1, \ldots, Y_m) = \sum_{i=1}^{n} x_i p_i(Y_1, \ldots, Y_m)$. In the above x is fixed, and $x_1, \ldots, x_n$ are fixed coefficients (and not variables). Note that $p_x(E(i)) = x_i$ for every $i \in [n]$ and $x \in F^n$.

With each of the above linear spaces we associate a natural low-degree encoding.[11] Specifically, we use:

E1: Let $\mathbf{E1}(i)$ be the i-th vector in $\text{GF}(2)^m$ of Hamming weight *at most* d. A proof of validity of this encoding, that is, the existence of appropriate polynomials $p_1, \ldots, p_n$, appears in Appendix B.

E2: Let $\omega_0, \ldots, \omega_d$ be distinct field elements. Then, $\mathbf{E2}(i)$ is the i-th vector of the form $(\omega_{f_1}, \ldots, \omega_{f_m})$ such that $\sum_{j=1}^{m} f_j \leq d$. A proof of the validity of this encoding may be found in Appendix B.

E3: Let $(i_1, \ldots, i_d)$ be the d-digit base-ℓ representation of i (that is, $i = \sum_{j=1}^{d} i_j \ell^{j-1}$). Then, $\mathbf{E3}(i)$ is a concatenation of the length-ℓ unit vectors $e_{i_1+1}, e_{i_2+1}, \ldots, e_{i_d+1}$. The validity of this encoding follows by letting $p_i = Y_{1,i_1+1} \cdot \ldots \cdot Y_{d,i_d+1}$.

[10] Whenever the field F is chosen to be GF(2) then n is also the length of x in bits; otherwise, n might actually be smaller.

[11] While the specific encoding being employed will usually not matter, in some cases it can make a difference. This is the case for the protocols in Section 5.3. Also, note that the definition of encoding does not require that the computation of $E(i)$ from i and the computation of p_x from x can be done efficiently; nevertheless, the encodings used in this work all have this property. This is important for making sure that we obtain computationally efficient PIR protocols.

Linear secret-sharing scheme. Denoted by L. The following t-private schemes will be useful.

$\mathbf{CNF}$: The k-player t-private CNF scheme from Definition 3.3. We will use the notation $\mathbf{CNF}_{k,t}$ whenever we want to emphasize the underlying parameters k and t.

$\mathbf{Shamir}$: The k-player t-private Shamir scheme from Definition 3.2. Again, we use $\mathbf{Shamir}_{k,t}$ to emphasize the parameters k and t.

$\mathbf{OptCNF}$: A slight optimization of the CNF scheme, whose details will be discussed in Section 7.

Simultaneous messages communication protocol (abbreviated SM protocol). The fourth and most crucial building block is a communication protocol for the following promise problem, which generalizes the scenario described in Section 2.2 and is defined by the instantiations of the previous components V and L. The protocol, denoted P, involves a user $\mathcal{U}$ and k servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$.

- *User's inputs:* Valid L-shares $y^1, \ldots, y^k$ of a point $y \in F^m$. (That is, the k vectors $y^1, \ldots, y^k$ can be obtained by applying L to each entry of y, and collecting the shares of each player.)
- *Servers' inputs:* All k servers hold a polynomial $p \in V$. In addition, each $\mathcal{S}_j$ holds the share vector y^j.
- *Communication pattern:* Each server $\mathcal{S}_j$ sends a single message to $\mathcal{U}$ based on its inputs p, y^j. We let β_j denote a bound on the length of the message sent by $\mathcal{S}_j$.
- *Output:* $\mathcal{U}$ should output $p(y)$.

We will sometimes need to rely on the additional promise that $y = E(i)$ for some $i \in [n]$ and some given encoding function E. In such cases, the input specification for the SM protocol will include E as well.

We say that the *probe complexity* of an SM protocol is c if the user can compute its output by probing at most c bits from each message (whose location may depend on its input).

In Section 5 we will present several SM protocols, corresponding to different choices of polynomial space V, secret-sharing scheme L, and encoding E.

4.2 Putting the Pieces Together

A 4-tuple (V, E, L, P) instantiating the above 4 primitives uniquely defines a PIR protocol $\mathrm{PIR}(V, E, L, P)$. The protocol proceeds as follows.

- $\mathcal{U}$ lets $y = E(i)$.
- $\mathcal{U}$ shares y according to L among the k servers. Let y^j denote the vector of shares received by $\mathcal{S}_j$.
- Each server $\mathcal{S}_j$ lets $p = p_x$ (as determined by the encoding E), and sends a message to $\mathcal{U}$ as specified by protocol P on inputs (p, y^j).
- $\mathcal{U}$ reconstructs $x_i = p(y)$ by applying the output computation function specified in P to $y^1, \ldots, y^k$ and the k messages it received.

The following lemma summarizes some easily verifiable properties of the above protocol.

Lemma 4.1 $\mathrm{PIR}(V, E, L, P)$ *is a t-private k-server PIR protocol, in which the user sends $m\ell_j$ field elements to each server $\mathcal{S}_j$ and receives β_j bits in return (where ℓ_j is the share size defined by L and β_j is the length of message sent by $\mathcal{S}_j$ in P). Moreover, the probe complexity of* $\mathrm{PIR}(V, E, L, P)$ *is the same as that of P.*

Note that the only information that a server gets is a share of the encoding $E(i)$; the t-privacy of the secret-sharing scheme ensures that any collusion of t servers learns nothing about i from their shares. For the query complexity, recall that $y = E(i) \in F^m$ and that the user shares each of the m coordinates of y independently. Thus, the share size of server $\mathcal{S}_j$ is indeed $m\ell_j$, where ℓ_j is the share size defined by L for sharing one coordinate (field element).

Remark 4.2 Some perspective concerning a typical choice of parameters is in place. We will usually view k (the number of servers) and t (the privacy threshold) as being constant, and the encoding length m (or n on which it depends) as the main complexity parameter. In such a setting each share size ℓ_j is also constant, and so the query complexity of $\mathrm{PIR}(V, E, L, P)$ is $O(m)$. If d is constant then, for any of the three vector spaces $\mathbf{V1}, \mathbf{V2}, \mathbf{V3}$, letting $m = O(n^{1/d})$ suffices to meet the dimension requirements. Thus, in the typical case where d, t, k are all constants, the length of the queries in $\mathrm{PIR}(V, E, L, P)$ is $O(n^{1/d})$ and the length of the answers is determined by P.

Remark 4.3 In principle, the SM component in our construction could be replaced by a more general interactive protocol. However, there is yet no evidence that such an additional interaction may be helpful. Moreover, in defining an interactive variant of the fourth primitive one would have to take special care that the privacy requirement is not violated by the interaction. In the current non-interactive framework, the privacy property is guaranteed by the mere use of a t-private secret-sharing scheme.

Remark 4.4 For the privacy of the PIR protocol, it is sufficient to use a t-private "distribution scheme" instead of a secret sharing scheme, where a t-private distribution scheme guarantees that every set of at most t players cannot reconstruct the secret, however sets of size $t + 1$ might not be able to reconstruct the secret. For example, the PIR protocols based on the balancing technique of [15] do not secret-share $E(i)$. All the protocols we describe in this paper use a secret-sharing scheme.

5 Simultaneous Messages Protocols

In this section we describe SM protocols corresponding to most useful combinations of V, L, and E. In Section 6 we will turn them into PIR protocols by choosing appropriate parameters. Some additional protocols are postponed to Section 7.

5.1 Protocol $\mathbf{P1}$: SM Protocol for Main PIR Family

Protocol $\mathbf{P1}$ will be used to obtain our main family of PIR protocols (see Section 6.1). It may be viewed as a natural generalization of the protocol from [4]. The inputs for $\mathbf{P1}$ are defined by the polynomial space $\mathbf{V1} = F_d[Y_1, \ldots, Y_m]$, where $F = \mathrm{GF}(2)$, and the secret-sharing scheme $\mathbf{CNF}$.

Lemma 5.1 *For $V = \mathbf{V1}_{d,m}$ and $L = \mathbf{CNF}_{k,t}$, there exists an SM protocol $\mathbf{P1}$ with message length $\beta_j = \Lambda(m, \lfloor dt/k \rfloor)\binom{k-1}{t-1}^{\lfloor dt/k \rfloor}$.*

Proof: Let $y = \sum_{T \in \binom{[k]}{t}} y_T$ be an additive sharing of y induced by the CNF scheme, such that the input y^j of $\mathcal{S}_j$ is $(y_T)_{j \notin T}$. The servers' goal is to communicate $p(y) = p(\sum_T y_T)$ to $\mathcal{U}$.

Let $\tilde{Y} = (Y_{T,b})_{T \in \binom{[k]}{t}, b \in [m]}$. Each variable $Y_{T,b}$ corresponds to the input bit $(y_T)_b$, whose value is known to all servers $\mathcal{S}_j$ such that $j \notin T$. Define a $\binom{k}{t}m$-variate polynomial $q(\tilde{Y}) \stackrel{\text{def}}{=} p\left(\sum_T Y_{T,1}, \ldots, \sum_T Y_{T,m}\right)$. Note that q has the same degree as p, and $q((y_{T,b})_{T \in \binom{[k]}{t}, b \in [m]}) = p(y)$. We consider the explicit representation of q as the sum of monomials, and argue that for every monomial $Y_{T_1,b_1} Y_{T_2,b_2} \ldots Y_{T_{d'},b_{d'}}$ of degree $d' \leq d$ there exists some $j \in [k]$ such that at most $\lfloor dt/k \rfloor$ variables $Y_{T,b}$ with $j \in T$ appear in the monomial. Indeed, consider the multiset $T_1 \cup T_2 \cup \ldots \cup T_{d'}$. Since this multiset contains $d't \leq dt$ elements, there must be some $j \in [k]$ that occurs at most $\lfloor dt/k \rfloor$ times. We partition the monomials of q into k polynomials $q_1, \ldots, q_k$ such that q_j contains only monomials in which the number of the variables $Y_{T,b}$ with $j \in T$ is at most $\lfloor dt/k \rfloor$, and each monomial of q is in exactly one polynomial q_j. Therefore $q(\tilde{Y}) = \sum_{j=1}^{k} q_j(\tilde{Y})$.

We are now ready to describe the protocol $\mathbf{P1}$. Denote by $\bar{Y}^j$ the set of variables whose values are *unknown* to the server $\mathcal{S}_j$ (that is, $\bar{Y}^j = (Y_{T,b})_{T \in \binom{[k]}{t}, j \in T, b \in [m]}$) and by $\bar{y}^j$ the corresponding values. Each $\mathcal{S}_j$ substitutes the values y^j of the variables it knows in q_j to obtain a polynomial $\hat{q}_j(\bar{Y}^j)$. The message of server $\mathcal{S}_j$ is a description of $\hat{q}_j$. The user, who knows the assignments to all variables, reconstructs $p(y)$ by computing $\sum_{j=1}^{k} \hat{q}_j(\bar{y}^j) = q((y_T)_{T \in \binom{[k]}{t}}) = p(y)$.

Communication complexity. Recall that $\hat{q}_j$ is a degree-$\lfloor dt/k \rfloor$ multivariate polynomial with $m\binom{k-1}{t-1}$ variables. By the definition of q, not all monomials are possible: no monomial contains two variables $Y_{T_1,b}$ and $Y_{T_2,b}$ for some $b \in [m]$ and $T_1 \neq T_2$. Thus, to describe a possible monomial of $\hat{q}_j$ we need, for some $w \in \{0, \ldots, \lfloor dt/k \rfloor\}$, to choose w indices in $[m]$ and w sets of size t that contain j. Therefore, the number of possible monomials of $\hat{q}_j$ is at most $\sum_{w=0}^{\lfloor dt/k \rfloor} \binom{m}{w} \binom{k-1}{t-1}^w \leq \Lambda(m, \lfloor dt/k \rfloor) \binom{k-1}{t-1}^{\lfloor dt/k \rfloor}$. Since each coefficient is from GF(2), the length of each message is as promised. □

Remark 5.2 To define the protocol $\mathbf{P1}$ precisely, one should specify a concrete partition of the polynomial q into the polynomials $q_1, \ldots, q_k$ (in particular, an assignment for the monomials that can be assigned to more than one polynomial). More generally, instead of *partitioning* the monomials, one may break q to any k polynomials which add up to q and satisfy the requirement utilized above. Such a generalization will be used in Section 5.3.1. However, for the purpose of Lemma 5.1, any partition of the monomials as described above is sufficient.

Remark 5.3 An important question is whether the communication complexity of the protocol $\mathbf{P1}$ is optimal. By extending a lower bound argument from [4] it is possible to show that, indeed, protocol $\mathbf{P1}$ is essentially optimal [23]. This is true even when given the additional promise that y is a valid $\mathbf{E1}$-encoding of some $i \in [n]$.

5.1.1 Reducing the Probe Complexity of $\mathbf{P1}$ for $t = 1$

Protocol $\mathbf{P1}$ requires the user to read all of the received bits. Next, we show that when $t = 1$ and given the promise that y is a legal encoding, i.e. $y = \mathbf{E1}(i)$ for some $i \in [n]$, then it is possible to modify $\mathbf{P1}$ so that the user needs to read much fewer bits. This is done without changing the communication complexity of the protocol.

Lemma 5.4 *For $V = \mathbf{V1}_{d,m}$, $E = \mathbf{E1}$, and $L = \mathbf{CNF}_{k,1}$, there exists an SM protocol $\mathbf{P1}'$ with message length $\Lambda(m, \lfloor d/k \rfloor)$ and probe complexity $\Lambda(d, \lfloor d/k \rfloor)$.*

Proof: Let $y = y_1 + \ldots + y_k$ be an additive sharing of $y = \mathbf{E1}(i)$, such that the input of each server $\mathcal{S}_j$ includes all pieces y_h with $h \neq j$. Similarly to Protocol $\mathbf{P1}$, we let each $\mathcal{S}_j$ compute a degree-$\lfloor d/k \rfloor$ polynomial $\hat{q}_j(Y_j)$, such that $\sum_{j=1}^{k} \hat{q}_j(y_j) = p(y)$. The key observation is that when $t = 1$, each value $\hat{q}_j(y_j)$ can also be expressed as the value of a degree-$\lfloor d/k \rfloor$ polynomial $q'_j(Y)$, known to $\mathcal{S}_j$, on the point y (rather than on y_j): the polynomial q'_j is defined as $q'_j(Y) = \hat{q}_j(Y - \sum_{h \neq j} y_h)$.

We can now rely on the promise that the Hamming weight of y is at most d to reduce the probe complexity. Similarly to $\mathbf{P1}$, each server $\mathcal{S}_j$ expresses $q'_j(Y)$ as a sum of monomials, and sends to the user the $\Lambda(m, \lfloor d/k \rfloor)$ coefficients (a single bit each). To evaluate $q'_j(y)$, the user only needs to read the coefficients of those monomials whose variables correspond to non-zero entries of y. Since y has at most d nonzero entries and the degree q' is at most $\lfloor d/k \rfloor$, there are at most $\Lambda(d, \lfloor d/k \rfloor)$ monomials involving the corresponding variables, and thus the probe complexity is as promised. □

The case $d = 2k - 1$ is the most useful one for constructing efficient 1-private PIR protocols. In this case it is possible to further reduce the probe complexity of $\mathbf{P1}'$ by one bit.

Lemma 5.5 *Suppose that d is odd and $\lfloor d/k \rfloor = 1$. Then, for $V = \mathbf{V1}_{d,m}$, $E = \mathbf{E1}$, and $L = \mathbf{CNF}_{k,1}$, there exists an SM protocol $\mathbf{P1}''$ with message length $\Lambda(m, \lfloor d/k \rfloor) = m + 1$ and probe complexity d.*

Proof: Protocol $\mathbf{P1}''$ will proceed similarly to $\mathbf{P1}'$ with the following modification: in the message sent by $\mathcal{S}_j$, the bit representing the free coefficient of q'_j is added (modulo 2) to each of the m other coefficients. The user, holding a point $y \in \mathrm{GF}(2)^m$ of Hamming weight $d' \leq d$, can evaluate $q'_j(y)$ as follows. If d' is odd, then $q'_j(y)$ is equal to the exclusive-or of the d' modified coefficients whose positions correspond to the nonzero entries of y. If d' is even (hence $d' < d$) then the user takes the exclusive-or of the same d' bits as above together with the free coefficient. In each of the two cases, the user reads at most d bits as required. □

5.2 Protocol $\mathrm{P2}$: SM Protocol for Binary PIR

Protocol $\mathbf{P2}$ is useful for the construction of efficient PIR protocols with short answers (see Section 6.2). Unlike protocol $\mathbf{P1}$, which can be used with any combination of the parameters k, d, t, the applicability of $\mathbf{P2}$ is restricted to the case $k > dt$. That is, $k = dt + 1$ is the minimal sufficient number of servers. The first part of the following lemma is implicit in [5, 6, 15] and a special case of the second part is implicit in [18, 19].

Lemma 5.6 *For $V = \mathbf{V2}_{d,m}$ and $L = \mathbf{Shamir}_{k,t}$, and assuming that $k > dt$ and $|F| > k$, there exists an SM protocol $\mathbf{P2}$ in which each server sends a single element of F. Moreover, given the promise that $p(y) \in F'$ for some subfield F' of F, it suffices for each server to send a single element of F'.*

Proof: Recall that in the definition of Shamir's scheme, a field element s is shared by evaluating a degree-t polynomial, whose free coefficient is s, on k distinct points ω_j, $1 \leq j \leq k$. Thus, in the setting of $\mathbf{P2}$ the user holds m univariate degree-t polynomials $q_1, \ldots, q_m$ such that $y = (q_1(0), \ldots, q_m(0))$, and $y^j = (q_1(\omega_j), \ldots, q_m(\omega_j))$ for $j \in [k]$. All servers hold a degree-d m-variate polynomial $p(Y)$ over F, and each server $\mathcal{S}_j$ holds the share y^j of y. The goal of the servers is to communicate the value $p(y)$ to $\mathcal{U}$ using a single element of F per server, or a single element of F' given the promise that $p(y) \in F'$.

We first describe the protocol in which each server sends a single element of F. In this case, the message sent by $\mathcal{S}_j$ in Protocol $\mathbf{P2}$ is $m_j = p(y^j)$. Note that the points (ω_j, m_j) lie on the graph of the degree-dt univariate polynomial $q(Z) \stackrel{\text{def}}{=} p(q_1(Z), \ldots, q_m(Z))$, and that $q(0) = p(y)$. Since $k > dt$, the user can reconstruct $q(Z)$ by interpolation and evaluate $q(0) = p(y)$. Specifically, there exist interpolation coefficients $c_1, \ldots, c_k$ such that $p(y)$ can be reconstructed from the messages $m_1, \ldots, m_k$ by applying the fixed linear combination $\sum c_j m_j$.

We now describe how to reduce the messages to be elements of a subfield F' given the promise that $p(y) \in F'$. Suppose first that each server modifies its message to $m'_j = c_j m_j$. Then, $\mathcal{U}$ may reconstruct $p(y)$ by *adding* the k messages m'_j it receives. As our final step, we use a field homomorphism $H : F \to F'$ such that $H(\alpha) = \alpha$ for all $\alpha \in F'$. (Since H is a homomorphism, for any $a, b \in F$ it holds that $H(a+b) = H(a) + H(b)$.) Now, instead of sending m'_j, the j-th server will send the F'-element $H(m'_j)$. From the properties of H and from the promise that $p(y) \in F'$ we may conclude that by adding the k answers (over F') the correct value $p(y)$ is obtained. $\square$

A special case of interest is when $F' = \mathrm{GF}(2)$ and F is a sufficiently large extension field of F'. In this case, each message in the SM protocol consists of a single bit. Note that when $k > dt$ the messages in Protocol $\mathbf{P1}$ are also one-bit long. However, for this choice of parameters $\mathbf{P2}$ is superior to $\mathbf{P1}$ as it relies on the (ideal) Shamir secret-sharing scheme, whereas $\mathbf{P1}$ relies on the highly redundant CNF scheme.

We do not know (and view it as an interesting problem) to prove nontrivial upper or lower bounds on the SM complexity in the $\mathbf{P2}$ setting when $k \le dt$. The lower bound proof technique of [4] and its generalizations from [33, 28] do not seem to apply to this setting. Good upper bounds could be used to eliminate the $\binom{k}{t}$ factors in our t-private constructions.

5.3 Protocol P3: SM Protocol for Probe-Efficient PIR

Similarly to protocols $\mathbf{P1}'$ and $\mathbf{P1}''$, the aim of protocol $\mathbf{P3}$ is to minimize the probe complexity while maintaining a low communication complexity. The inputs for $\mathbf{P3}$ are defined by $V = \mathbf{V3}$, $E = \mathbf{E3}$, and $L = \mathbf{CNF}$. A special case of this protocol is implicit in the 2-server PIR protocol from [15].

Another similarity between $\mathbf{P3}$ to $\mathbf{P1}'$ and $\mathbf{P1}''$ is that its version for the case $t = 1$ does not smoothly generalize to larger values of t. However, in contrast to the latter two protocols, we will use the special structure of the $\mathbf{E3}$ encoding to obtain some useful generalizations of $\mathbf{P3}$ for the case $t > 1$. [12]

We will first focus on the simpler case where $t = 1$ and later address the general case.

Lemma 5.7 *For $V = \mathbf{V3}_{d,m}$, $E = \mathbf{E3}$, and $L = \mathbf{CNF}_{k,1}$, there exists an SM protocol $\mathbf{P3}$ with message length $\ell^{\lfloor d/k \rfloor}\binom{d}{\lfloor d/k \rfloor}$ and probe complexity $\binom{d}{\lfloor d/k \rfloor}$.*

Proof: We first explicitly specify the inputs to the protocol. Let p be a polynomial in $\mathbf{V3}_{d,m}$, and let $y \in \{0,1\}^{\ell d}$ be an encoding of some index i under the $\mathbf{E3}$ encoding, that is, $y = w_1 \circ \ldots \circ w_d$, where $\circ$ denotes concatenation and each w_h is some unit vector from the space F^ℓ. (By default we let $F = \mathrm{GF}(2)$, but the following discussion may apply to an arbitrary finite field.) Furthermore, let $w_h = \sum_{a=1}^{k} w_{a,h}$ be an additive sharing of w_h induced by the 1-private $\mathbf{CNF}$ scheme. We will refer to each ℓ-tuple $w_{a,h}$ as a *piece* of w_h. The input y^j of $\mathcal{S}_j$ is $(w_{a,h} : a \in [k] \setminus \{j\}, h \in [d])$. The servers' goal is to efficiently communicate $p(y)$ to the user such that the user will need to read only $\binom{d}{\lfloor d/k \rfloor}$ bits from each message.

Recall that $p \in \mathbf{V3}_{d,m}$, i.e., each of its monomials contains exactly one variable from each of the d segments. It can therefore be viewed as a *multi-linear* function of d vectors in F^ℓ. We can thus write:

$$p(y) \quad = \quad p(w_1 \circ \ldots \circ w_d)$$

[12] In fact, one can construct a similar protocol based on the encoding $\mathbf{E1}$ (rather than $\mathbf{E3}$) whose probe complexity and communication complexity are inferior to those of the protocol presented below only by poly-logarithmic factors. Informally speaking, this is done by "covering" $\mathbf{E1}$ using a poly-logarithmic number of copies of $\mathbf{E3}$ and using the solution for $\mathbf{E3}$ described below.

$$= \quad p((\sum_{a=1}^{k} w_{a,1})\circ \ldots \circ(\sum_{a=1}^{k} w_{a,d}))$$
$$= \sum_{a_1\in[k],\ldots,a_d\in[k]} p(w_{a_1,1}\circ \ldots \circ w_{a_d,d}). \quad (1)$$

Each expression $p(w_{a_1,1}\circ \ldots \circ w_{a_d,d})$ in the sum (1) will be referred to as a *term*, and will be denoted from now on by the d-tuple $(a_1, \ldots, a_d)$. Notice that each term evaluates to a single field element (a single bit by default). Moreover, the user's goal of learning $p(y)$ may now be reduced to learning the *sum* of all k^d terms.

Each term contains d segments and each segment is missed by one server. Thus, if $k > d$ then every term can be evaluated by at least one server. In this case, we partition the responsibility for evaluating the k^d terms among the servers, where each server $\mathcal{S}_j$ is only responsible for evaluating terms $(a_1, \ldots, a_d)$ such that $a_h \neq j$ for $h = 1, \ldots, d$. Each server sums the values of the terms it is responsible for and sends a single bit to the user.

If $k \leq d$, then, similarly to Protocol $\mathbf{P1}$, we partition the responsibility for evaluating the k^d terms among the servers, where each server $\mathcal{S}_j$ is only responsible for evaluating terms $(a_1, \ldots, a_d)$ such that $a_h = j$ for at most $\lfloor d/k \rfloor$ indices h. Server $\mathcal{S}_j$, missing at most $\lfloor d/k \rfloor$ pieces in each of the terms assigned to it, tries to guess them in a clever way. This is done using the knowledge that each w_h is promised to be a unit vector. Therefore, $w_{j,h} = e_i - \sum_{a\in[k]\setminus\{j\}} w_{a,h}$ for some unit vector $e_i \in \{0,1\}^\ell$, and hence $\mathcal{S}_j$ can compute a list of ℓ possible values (corresponding to the ℓ unit vectors) for each missing piece $w_{j,h}$. This already implies a protocol where each server sends $k^d \ell^{\lfloor d/k \rfloor}$ bits: There are k^d terms that need to be evaluated. For each such term $(a_1, \ldots, a_d)$, the server that is responsible for it evaluates p using the pieces that it knows and each of the guesses for the pieces it does know; there are at most $\ell^{\lfloor d/k \rfloor}$ such guesses. The server sends this list of guesses to the user. The user, who knows the values of all pieces, can compute the true value of each term.

We next optimize this protocol. Instead of sending a list of length $\ell^{\lfloor d/k \rfloor}$ for each of the k^d terms, each server sends only $\binom{d}{\lfloor d/k \rfloor}$ lists, one for each set $A \in \binom{[d]}{\lfloor d/k \rfloor}$. This is done by assigning each term $(a_1, \ldots, a_d)$ which is under the responsibility of $\mathcal{S}_j$ to some set $A \in \binom{[d]}{\lfloor d/k \rfloor}$ such that $\{h : a_h = j\} \subseteq A$. Now, each of the $\ell^{\lfloor d/k \rfloor}$ possible guesses of $(w_{a_h,h})_{h\in A}$ allows $\mathcal{S}_j$ to uniquely determine all terms which are assigned to A, and therefore also their sum. Thus, $\mathcal{S}_j$ can send, for each of the $\ell^{\lfloor d/k \rfloor}$ guesses and each set A, a single bit containing this sum. Finally, note that the user needs to read only one bit from each list (corresponding to the correct guess), giving a total of $\binom{d}{\lfloor d/k \rfloor}$ probed bits per server. $\square$

5.3.1 Probe-efficient protocols with $t > 1$

Towards handling arbitrary values of t, we will first attempt at a straightforward generalization of the previous approach. As before, let $y = w_1\circ \ldots \circ w_d$, where each w_h is some unit vector from F^ℓ. Denote the pieces (i.e., additive shares) of w_h by $w_{T,h}$, where $T \in \binom{[k]}{t}$. Recall that each server $\mathcal{S}_j$ receives all pieces $w_{T,h}$ such that $j \notin T$. Generalizing Eq. (1), the servers' goal is to efficiently communicate to the user

$$p(w_1\circ \ldots \circ w_d) \quad = \quad p((\sum_{T\in\binom{[k]}{t}} w_{T,1})\circ \ldots \circ(\sum_{T\in\binom{[k]}{t}} w_{T,d}))$$
$$= \sum_{(T_1,\ldots,T_d)\in\binom{[k]}{t}^d} p(w_{T_1,1}\circ w_{T_2,2}\circ \ldots \circ w_{T_d,d})$$

Again, each expression $p(w_{T_1,1}\circ w_{T_2,2}\circ \ldots \circ w_{T_d,d})$ in the last sum will be referred to as a term, and denoted by the d-tuple of sets $(T_1, \ldots, T_d)$. The user's goal of learning $p(w_1\circ \ldots \circ w_d)$ may now be reduced to learning the sum of all $\binom{k}{t}^d$ terms.

As before, we try to assign each of the terms to a server which can come up with a short list of candidates for its value. We previously relied on the fact that a server could come up with a list of ℓ candidate values for each missing piece. However, this is not possible in general. Indeed, in the t-private CNF scheme with $t > 1$, a server $\mathcal{S}_j$ misses more than one piece of each secret w_h. It is thus no longer possible to compute the value of a missing piece even when given a correct guess for the value of the shared unit vector w_h.

We therefore take a somewhat different approach. Since we cannot require a server to guess the value of an unknown piece $w_{T_h,h}$, we will try to directly utilize the promise that the *sum* $\sum_{T\in\binom{[k]}{t}} w_{T,h}$ is a length-ℓ unit vector, and can thus be efficiently guessed. Note that for any server $\mathcal{S}_j$, this sum involves both known and unknown pieces. Moreover, if $j \notin T_{h'}$ for each $h' \in [d] \setminus \{h\}$ (so that $\mathcal{S}_j$ holds all pieces $w_{T_{h'},h'}$) then $\mathcal{S}_j$ can easily obtain a list of ℓ candidates for the value of the *sum of terms*

$$\sum_{T\in\binom{[k]}{t}} (T_1, \ldots, T_{h-1}, T, T_{h+1}, \ldots, T_d) = \sum_{T\in\binom{[k]}{t}} p(w_{T_1,1}\circ \ldots \circ w_{T_{h-1},h-1} \circ w_{T,h} \circ w_{T_{h+1},h+1} \circ \ldots \circ w_{T_d,d}).$$

We will refer to such a sum as a *block* and denote it by $(T_1, \ldots, T_{h-1}, *, T_{h+1}, \ldots, T_d)$ (where the wild-card symbol '*' represents summation over all $\binom{k}{t}$ pieces). This notion will be later made more general and more precise, but for the time being a block should be thought of as a linear combination of terms which can be efficiently communicated with a low probe complexity.

We now consider the vector space over F whose elements are all possible linear combination of terms. (Here and in the following, terms should be treated by default as *syntactic* objects which form the standard basis of the vector space of their linear combinations.) Note that the dimension of this space is $\binom{k}{t}^d$ and both the blocks and the sum of all terms are vectors in this space. The key observation is that if the set of available blocks happens to *span* the sum of all terms, then the user can learn the correct output from the values of the blocks. When $F = \mathrm{GF}(2)$, the above spanning property is equivalent to the existence of a set of blocks in which each term occurs an odd number of times.

In the following we formalize the above discussion and obtain necessary and sufficient conditions for the spanning property to hold. We start by defining a more general class of blocks.

Definition 5.8 *Let $0 \le \delta \le d$ be an integer, $h_1, h_2, \ldots, h_{d-\delta}$ be distinct indices from $[d]$, and $S_1, \ldots, S_{d-\delta} \in \binom{[k]}{t}$ be $d-\delta$ sets whose union does* not *cover $[k]$. The δ*-block *defined by $h_1, \ldots, h_{d-\delta}$ and $S_1, \ldots, S_{d-\delta}$ is the sum of all terms $(T_1, \ldots, T_d)$ such that $T_{h_j} = S_j$ for $j = 1, \ldots, d-\delta$. Such a block will be denoted by a d-tuple whose $d-\delta$ entries $h_1, \ldots h_{d-\delta}$ contain (respectively) the sets $S_1, \ldots, S_{d-\delta}$, and whose remaining δ entries contain the wild-card symbol '*'.*

Note that a δ-block is a summation of $\binom{[k]}{t}^\delta$ terms, whose names lie in some δ-dimensional subspace of the d-dimensional space $\binom{[k]}{t}^d$ of term names. However, not every d-tuple made of sets and wild-card symbols represents a valid block. Some examples follow.

Example 5.9 *For $k = 4, d = 3$, and $t = 2$, the triplet $(\{2,4\}, \{2,4\}, \{1,2\})$ is a valid 0-block, the triplet $(\{2,4\}, *, \{1,2\})$ is a valid 1-block, and $(\{3,4\}, *, *)$ is a valid 2-block. However, $(\{2,4\}, \{3,4\}, \{1,2\})$ and $(\{2,4\}, *, \{1,3\})$ are not valid blocks, since their sets cover $[k]$.*

We now define two useful notions related to the expressive power of a set of blocks in a given term space. For convenience, we identify each block with the set of terms it involves.

Definition 5.10 *Fix t, d, k. Let $\mathcal{T} \subseteq \binom{[k]}{t}^d$ be a set of terms and $\mathcal{B}$ be a set of blocks. We say that $\mathcal{B}$* covers *$\mathcal{T}$, if any term $\tau \in \mathcal{T}$ is included in some block from $\mathcal{B}$. We say that $\mathcal{B}$* spans the sum of terms in *$\mathcal{T}$ (or* spans *$\mathcal{T}$ for short), if there exists a linear combination of the blocks in $\mathcal{B}$ resulting in the sum $\sum_{\tau \in \mathcal{T}} \tau$ (where each block is viewed as a linear combination, with coefficients from F, of terms from the entire space).*

Using the above terminology, the protocol of Lemma 5.7 utilized the fact that every term was spanned by the set of 0-blocks and 1-blocks held by some server.

Remark 5.11 (On the role of the field F) Notice that whether or not a given set of blocks spans a given set of terms may depend on the underlying field F. For instance, if there exists a set of blocks such that every term is covered by *exactly* 4 blocks from this set, then the sum of all terms is clearly spanned over GF(3), but is not necessarily spanned over GF(2). Nevertheless, we use $F = \mathrm{GF}(2)$ throughout this section since this is sufficient (and at one point necessary) for our upper bounds.

We now generalize the protocol of Lemma 5.7 to use a general spanning condition.

Lemma 5.12 *Fix k, d, t and an integer $0 \le \delta < d$, and let $\mathcal{B}_\delta$ be the set of all δ'-blocks with $0 \le \delta' \le \delta$. Suppose that $\mathcal{B}_\delta$ spans the sum of all terms in $\mathcal{T} = \binom{[k]}{t}^d$. Then, for $V = \mathbf{V3}_{d,m}$, $E = \mathbf{E3}$, and $L = \mathbf{CNF}_{k,t}$, there exists an SM protocol $\mathbf{P3}$ with message length $\ell^\delta \binom{d}{\delta}$ and probe complexity $\binom{d}{\delta}$.*

Proof: The proof is similar to that of Lemma 5.7. However, while there we tried to span each term separately, here we will directly span the sum of all terms.

The protocol proceeds as follows. Each δ'-block $b \in \mathcal{B}$ is assigned to some server whose index does not occur in any of the $d - \delta'$ sets of b. The block b is further assigned to some set $A \in \binom{[d]}{\delta}$ which contains all of the wild-card coordinates of b. Let $\mathcal{B}_{j,A}$ denote the set of all blocks in $\mathcal{B}_\delta$ assigned to $\mathcal{S}_j$ and A.

The message sent by each server $\mathcal{S}_j$ consists of $\binom{d}{\delta}$ lists, one for each set $A \in \binom{[d]}{\delta}$. The list for the set $A = \{h_1, \ldots, h_\delta\}$ contains ℓ^δ field elements, each corresponding to a different choice of length-ℓ unit vectors for the wild-cards in coordinate set A. Specifically, suppose that $h_1 < h_2 < \ldots < h_\delta$. For each δ'-block $b \in \mathcal{B}_{j,A}$ and each position $(i_1, \ldots, i_\delta)$ of the list, server $\mathcal{S}_j$ evaluates p on the pieces specified in the block, where each wild-card in position h_a is replaced by the unit vector e_{i_a}. This defines a separate list of length ℓ^δ for each block $b \in \mathcal{B}_{j,A}$. (Note that when $\delta' < \delta$, the list will contain duplicate entries.) Let $L_{j,A,b}$ denote this list. As a final optimization, all the lists $L_{j,A,b}$ are linearly combined to the single list $L_{j,A} = \sum_{b \in \mathcal{B}_{j,A}} c_b L_{j,A,b}$ where $\sum_{b \in \mathcal{B}_\delta} c_b b$ is some fixed linear combination of the blocks in $\mathcal{B}_\delta$ which gives the sum of all terms. (The existence of such a linear combination is guaranteed by the spanning premise.) The message sent by each server $\mathcal{S}_j$ will consist of all lists $L_{j,A}$, giving the promised message length.

Finally, the user, holding $i = (i_1, \ldots, i_d) \in [\ell]^d$, can compute $p(E(i))$ as the sum

$$\sum_{j=1}^{k} \sum_{\substack{A = \{h_1, \ldots, h_\delta\} \\ h_1 < h_2 < \ldots < h_\delta}} (L_{j,A})_{(i_{h_1}, i_{h_2}, \ldots, i_{h_\delta})}$$

This requires probing one bit from each of the $\binom{d}{\delta}$ lists in each message. □

To make use of Lemma 5.12, one should study the relation that the various parameters must satisfy in order for the spanning condition to hold. We will restrict our attention from now on to the case $\delta = 1$, and will view the parameters k, d, t as constants. The importance of this case is that it provides "balanced" PIR protocols, in which the length of the queries is roughly the same as that of the answers.[13] This motivates the study of the following question:

Question 5.13 *Given d and t, how large should k be so that the set $\mathcal{B}_1$ of 0-blocks and 1-blocks spans the sum of all terms?*

In the remainder of this section we will solely focus on this question. First, we introduce some notation.

Definition 5.14 *Let $k_c(d,t)$ denote the smallest integer k such that the term space $\binom{[k]}{t}^d$ is* covered *by its 0-blocks and 1-blocks,*[14] *and $k_s(d,t)$ denote the smallest k such that the sum of all terms is* spanned *by these blocks.*

An exact characterization of $k_c(d,t)$ and some simple bounds on $k_s(d,t)$ are given by the following two claims.

Claim 5.15 *For any positive integers d and t, $k_c(d,t)$ is the smallest integer greater than $dt/2$. That is,*

$$k_c(d,t) = \lfloor dt/2 \rfloor + 1.$$

Proof: We show that the term space $\binom{[k]}{t}^d$ is covered by 1-blocks if and only if $k > dt/2$. If $dt \geq 2k$, then it is possible to construct a term $\tau = (T_1, T_2, \dots, T_d)$ such that each element $j \in [k]$ occurs in at least two sets T_h (e.g., by greedily adding each element to a pair of vacant sets) and such a term cannot be covered by a 1-block. Conversely, if there exists a term $\tau = (T_1, T_2, \dots, T_d)$ which is not covered by any 1-block, then every element $j \in [k]$ must occur in at least two sets T_h, implying that $dt \geq 2k$. □

Claim 5.16 *For any positive integers d and t, and over any field F,*

$$k_c(d,t) \leq k_s(d,t) \leq dt + 1.$$

Proof: The fact that k_c is a lower bound on k_s is an immediate consequence of the definitions. The upper bound on $k_s(d,t)$ follows from the fact that the *0-blocks* alone are sufficient to cover the entire term space, and hence to span the sum of all terms, whenever $k > dt$. □

The protocol of Lemma 5.7 shows the lower bound on k_s provided by Claim 5.16 is tight for the case $t = 1$. However, this is not the case in general (see Appendix C). In the following we will improve the upper bound on $k_s(d,t)$ for $t > 1$. A first observation to make is that, in addition to the 0-blocks, there are other singleton sets of terms which can be spanned.

Lemma 5.17 *If a term $\tau = (T_1, T_2, \dots, T_d)$ includes a set T_h which is* disjoint *from all $d-1$ other sets, then $\{\tau\}$ is spanned.*

Proof: Let $\tau = (T_1, T_2, \dots, T_d)$ be such a term, with T_h satisfying $T_h \cap T_{h'} = \emptyset$ for any $h' \in [d] \setminus \{h\}$. The definition of valid blocks implies that:

- $(T_1, \dots, T_{h-1}, *, T_{h+1}, \dots, T_d)$ denotes a valid 1-block (since the $d-1$ sets in this d-tuple do not include any element of T_h, and thus cannot cover $[k]$); and

[13]When applying Lemma 5.12 with $\delta > 1$, the communication complexity of the SM protocol is at least quadratic in ℓ, whereas the length of the inputs (or PIR queries) is linear in ℓ.

[14]Since each 0-block is covered by (exactly d) 1-blocks, it is sufficient in the definition of k_c to consider 1-blocks.

- For any $T \in \binom{[k]}{t} \setminus \{T_h\}$, the d-tuple

$$(T_1, \ldots, T_{h-1}, T, T_{h+1}, \ldots, T_d)$$

denotes a valid 0-block (since $T_h \setminus T \neq \emptyset$ and each element of $T_h \setminus T$ is not covered by the d sets).

Finally, writing τ as:

$$\tau = (T_1, \ldots, T_{h-1}, *, T_{h+1}, \ldots, T_d) - \sum_{T \in \binom{[k]}{t} \setminus \{T_h\}} (T_1, \ldots, T_{h-1}, T, T_{h+1}, \ldots, T_d),$$

the lemma follows. □

We can now focus our attention on the terms which we do not know how to handle directly.

Definition 5.18 *A term $(T_1, T_2, \ldots, T_d)$ is said to be* bad*, if it is not a 0-block, and it does not include a set T_j disjoint from all others.*

Lemma 5.19 *Suppose that a term set $\mathcal{T}$ containing all bad terms (and possibly some other terms) can be spanned. Then the set of all terms can be spanned.*

Proof: The 0-blocks and the terms handled by Lemma 5.17 (replaced by their spanning sets of blocks) span $\binom{[k]}{t}^d \setminus \mathcal{T}$. □

We are now ready to prove upper bounds on $k_s(d,t)$. The first bound applies only to an *odd* d. The second bound is slightly weaker, but can be applied also when d is even.

Lemma 5.20 *For any positive integer $t \geq 1$ and* odd *integer $d \geq 3$,*

$$k_s(d,t) \leq \min\left(\lfloor dt - (d+t-3)/2 \rfloor \ , \ dt - t\right).$$

Proof: It suffices to show that when d is odd, and when either $k > dt - \frac{d+t-1}{2}$ or $k > dt - t - 1$, then the sum of all terms is spanned. Suppose k, d and t meet the above constraints. By Lemma 5.19, it suffices to show that some term set $\mathcal{T}$ containing all bad terms can be spanned. We now show that this is achieved by the block set $\mathcal{B}$ which includes all 1-blocks in which the element '1' occurs an *even* number of times.

Lemma 5.21 *For k, d, t constrained as above, each bad term is included in an odd number of blocks from the set $\mathcal{B}$.*

Proof: Suppose that this is not the case; that is, there exists a bad term $\tau = (T_1, \ldots, T_d)$ which is covered by an even number of blocks from $\mathcal{B}$. Now, consider the number of occurrences of '1' in τ. We continue with some observations that will help us prove the claim.

- If '1' does not occur at all, then τ is not bad, contradicting the assumptions.
- If '1' occurs exactly once, in the set T_h, then there is exactly one block from $\mathcal{B}$ covering τ. Specifically, the block obtained by "removing" T_h (i.e., replacing it with a wild-card) contains an even number of 1's (namely, 0 times), and it is a valid 1-block since its sets do not contain '1'. We conclude that this case as well contradicts the assumptions.

- If '1' occurs e times, where e is *even*: there are $(d-e)$ *candidates* for blocks in $\mathcal{B}$ covering τ, namely all those obtained by removing a set which does not contain '1'. Since d is odd and e is even, $d-e$ is odd. It follows that for τ to be covered by an even number of blocks from $\mathcal{B}$, at least one of the $d-e$ candidates must not be a valid 1-block, implying that some set T_z *which does not contain '1'* must be covered by all other sets.

- If '1' occurs o times, where o is odd and is greater than 1, then there are o candidates for blocks in $\mathcal{B}$ covering τ, namely all those obtained by removing a set which contains '1'. Again, it follows that for τ to be covered by an even number of blocks from $\mathcal{B}$ at least one of these candidates must not be a valid 1-block, implying that at least one of the (at least three) sets T_z which contain '1' must be covered by all other sets.

We deduce from the above cases that there must exist sets T_z, T_{p_1} and T_{p_2}, with *distinct* indices z, p_1, p_2, such that: (a) T_z is covered by the union of all other $d-1$ sets; and (b) $1 \in T_{p_1} \cap T_{p_2}$. Moreover, since τ is bad, we have: (c) every set T_h intersects some other set; and (d) the union of all d sets is $[k]$.

Now, what is the maximal k such that the above four conditions can be simultaneously satisfied by some term τ? Equivalently, what is the maximum size of the *union* of d sets $T_1, \ldots, T_d$ of size t each, such that conditions (a),(b), and (c) above are met? Denote this maximum union size by $k^*_{odd}(d,t)$. The following two propositions bound $k^*_{odd}(d,t)$ from above.

Proposition 5.22 $k^*_{odd}(d,t) \leq dt - t - 1$.

Proof: For any sets $T_1, \ldots, T_d$ meeting the above conditions (a) and (b), we have:

$$\begin{aligned} \left| \bigcup_{h \in [d]} T_h \right| &= \left| \bigcup_{h \in [d] \setminus \{z\}} T_h \right| \\ &\leq (d-1)t - 1 \\ &= dt - t - 1, \end{aligned}$$

where the first equality follows from condition (a) and the inequality from condition (b). □

Proposition 5.23 $k^*_{odd}(d,t) \leq dt - \frac{d+t-1}{2}$.

Proof: Suppose that $T_1, \ldots, T_d$ meet the above conditions (a) and (c). For each $w \in T_z$, let h_w be an index of some set other than T_z which includes w (where the indices h_w need not be distinct, and their existence is guaranteed by condition (a)). Let $Z = \{z\} \cup \{h_w : w \in T_z\}$. Next, for every $h \in [d] \setminus Z$, we select an element y_h which occurs both in T_h and in some other set (again, the y_h need not be distinct, and their existence is guaranteed by condition (c)). Finally, we group the sets with indices from $[d] \setminus Z$ according to their selected y_h. That is, for each $w \in [k]$ we define a "cluster" $G_w \stackrel{\text{def}}{=} \{h \in [d] \setminus Z : y_h = w\}$.

We analyze how the union size grows, when we first take all sets with indices from Z, then add each cluster G_w containing two or more sets, and finally add all clusters G_w containing a single set. The union of all sets from Z is no larger than $t|Z| - t$, since each element of T_z is counted there at least twice. When adding a cluster G_w with at least two sets, its $|G_w|$ sets contribute at most $t|G_w| - (|G_w| - 1) \leq (t - \frac{1}{2})|G_w|$ new elements to the union size. Finally, we add together all singleton clusters G_w. Suppose there are s of them. Then, their total contribution to the union size is no larger than $(t-1)s$, since the element y_h in each such unmatched set T_h either occurs in a previously added set, or occurs in some other singleton cluster (and will only be accounted for once). Altogether, we have that:

$$\begin{aligned}
\left|\bigcup_{h\in[d]} T_h\right| &\leq \left|\bigcup_{h\in Z} T_h\right| + \left|\bigcup_{h\in[d]\setminus Z} T_h\right| \\
&\leq (t|Z|-t) + (t-1/2)(d-|Z|) \\
&= dt - t + (|Z|-d)/2 \\
&\leq dt - t + (t+1-d)/2 \\
&= dt - (d+t-1)/2,
\end{aligned}$$

as required. □

We remark that the above analysis is tight; that is, k^*_{odd} can be shown to be *equal* to $\min(dt - t - 1, \left\lfloor dt - \frac{d+t-1}{2} \right\rfloor)$.

It may now be concluded that when d is odd and $k > k^*_{odd}(d,t)$, no bad term can be covered by an even number of blocks from $\mathcal{B}$; combining this with Propositions 5.22 and 5.23, Lemma 5.21 follows. □

Since all linear combinations are taken over $F = \mathrm{GF}(2)$, Lemma 5.21 implies that a set containing all bad terms can be spanned, which by Lemma 5.19 concludes the proof of Lemma 5.20. □

We now prove a slightly weaker bound on $k_s(d,t)$, which applies to *any* d and over any field F (although only an even d and $F = \mathrm{GF}(2)$ will be useful for our purpose). In fact, when $d \geq t+2$ this bound is identical to the previous one, and otherwise the difference between the bounds is 1.

Lemma 5.24 *For any positive integers d, t,*

$$k_s(d,t) \leq \min\left(\lfloor dt - (d+t-3)/2 \rfloor \ , \ dt - t + 1\right).$$

Proof: The proof goes along the lines of the proof of Lemma 5.20. Again, we define a set of blocks $\mathcal{B}$, and show that it is guaranteed to span some set containing all bad blocks whenever k is no smaller than the bound of Lemma 5.24. In this case, we let $\mathcal{B}$ be the set of all valid 1-blocks of the form $(*, T_2, \ldots, T_d)$. Clearly, every term can be covered by at most one block from $\mathcal{B}$. We show that every *bad* term is covered by exactly one block from $\mathcal{B}$.

Suppose that $\tau = (T_1, \ldots, T_d)$ is a bad term not covered by $\mathcal{B}$. It follows that the set T_1 is covered by the union of the other sets, implying that condition (a) from the proof of Lemma 5.20 holds with $h = 1$. Since τ is assumed to be bad, conditions (c) and (d) apply as well. Defining k^*_{even} analogously to k^*_{odd}, the proof of Proposition 5.23 can be used to show that $k^*_{even}(d,t) \leq dt - \frac{d+t-1}{2}$ (since only conditions (a),(c) and (d) are used in this proof). Finally, $k^*_{even}(d,t) \leq dt - t$, since for any sets $T_1, \ldots, T_d$ meeting condition (a) we have:

$$\left|\bigcup_{h\in[d]} T_h\right| = \left|\bigcup_{h\in[d]\setminus\{1\}} T_h\right| \leq dt - t.$$

□

Substituting the bounds from Lemmas 5.20 and 5.24 in Lemma 5.12, we conclude the following:

Corollary 5.25 *Suppose that k, d, t satisfy*

$$k \geq \min\left(\lfloor dt - (d+t-3)/2 \rfloor \ , \ dt - t + 1 - (d \bmod 2)\right).$$

Then, for $V = \mathbf{V3}_{d,m}$, $E = \mathbf{E3}$, and $L = \mathbf{CNF}_{k,t}$, there exists an SM protocol $\mathbf{P3}$ with message length $d\ell$ and probe complexity d.

6 Families of PIR Protocols

In this section we derive several explicit families of PIR protocols from the meta-construction.

6.1 Main PIR Family

Our main family of PIR protocols uses $\mathbf{V1}$, $\mathbf{E1}$, $\mathbf{CNF}$, and $\mathbf{P1}$, $\mathbf{P1}'$, or $\mathbf{P1}''$. Protocols from this family yield our main improvements over previous upper bounds. We start with the general result, which follows from Lemmas 4.1 and 5.1, and then consider some interesting special cases.

Theorem 6.1 *Let m and d be positive integers such that $\Lambda(m,d) \geq n$. Then, for any k, t such that $1 \leq t < k$, there exists a t-private k-server PIR protocol with query length $\binom{k-1}{t}m$ (per server) and answer length $\Lambda(m, \lfloor dt/k \rfloor)\binom{k-1}{t-1}^{\lfloor dt/k \rfloor}$ (per server).*

Proof: The condition $\Lambda(m,d) \geq n$ is sufficient (and necessary) to guarantee the existence of an encoding $\mathbf{E1}$ as required. We use the $\mathbf{CNF}_{k,t}$ secret-sharing scheme and thus the length of each share is $\binom{k-1}{t}m$. Finally, we use the SM protocol $\mathbf{P1}$. By Lemma 5.1, the length of the message of each server is $\Lambda(m, \lfloor dt/k \rfloor)\binom{k-1}{t-1}^{\lfloor dt/k \rfloor}$. □

The total communication is optimized by letting $d = \lfloor (2k-1)/t \rfloor$ and $m = \Theta(n^{1/d})$. Substituting these parameters into Theorem 6.1 and using the approximation of Λ, described in Appendix A, we get the following explicit bounds.

Corollary 6.2 *For any $1 \leq t < k$ there exist:*

1. *A t-private k-server PIR protocol with communication complexity $O(\frac{k^2}{t}\binom{k}{t}n^{1/\lfloor (2k-1)/t \rfloor})$.*
2. *A 1-private k-server PIR protocol with communication complexity $k^2((2k-1)!n)^{1/(2k-1)} + k + k^3 = O(k^3\, n^{1/(2k-1)})$.*
3. *A 1-private 2-server PIR protocol with communication complexity $4(6n)^{1/3} + 2 \approx 7.27n^{1/3}$.*

Proof: The first bound is obtained from Theorem 6.1 by letting $d = \lfloor (2k-1)/t \rfloor$ and $m = O(dn^{1/d})$. The second is obtained by letting $m = ((2k-1)!\, n)^{1/(2k-1)} + k$ and $d = 2k-1$, and the third by letting $m = (6n)^{1/3}$ and $d = 3$. □

Another interesting case, discussed and used in [10], is when the queries are of length $O(\log n)$. Such protocols are obtained by letting $d = \theta m$, where $0 < \theta \leq 1/2$ is some constant.

Corollary 6.3 *For any constant integers t, k (where $1 \leq t < k$) and constant $\epsilon > 0$, there exists a t-private k-server protocol with query length $O(\log n)$ and answer length $O(n^{t/k+\epsilon})$. More precisely, for any $0 < \theta \leq 1/2$ there is a protocol with query length $\binom{k-1}{t}(1/H(\theta) + o(1))\log n$ and answer length $n^{(H(\theta t/k) + \theta\frac{t}{k}\log\binom{k-1}{t-1})/H(\theta)+o(1)}$.*

Proof: The corollary follows by substituting $m = (1/H(\theta) + o(1))\log n$ and $d = \lfloor \theta m \rfloor$ in Theorem 6.1, and relying on the facts that $\lim_{\theta \to 0}\frac{H(\theta t/k)}{H(\theta)} = t/k$ and $\lim_{\theta\to 0}\frac{\theta}{H(\theta)} = 0$. □

As discussed in Section 1.2, a 1-private k-server PIR protocol with query length α and answer length β gives rise to a k-query LDC of length $k \cdot 2^{\alpha}$ over the alphabet $\Sigma = \{0,1\}^{\beta}$. If $\alpha = O(\log n)$, then the code length is polynomial. By substituting $t = 1$ in Corollary 6.3 we get:

Corollary 6.4 *For any constant integer $k \geq 2$ and constant $\epsilon > 0$, there is a* polynomial-length *k-query LDC over $\Sigma(n) = \{0,1\}^{\beta(n)}$, where $\beta(n) = O(n^{1/k+\epsilon})$.*

In the case $t = 1$, protocols $\mathbf{P1}'$ and $\mathbf{P1}''$ can be used to obtain probe-efficient variants of Theorem 6.1.

Theorem 6.5 *Let m and d be positive integers such that $\Lambda(m,d) \geq n$. Then, for any k there exists a 1-private k-server PIR protocol with query length $(k-1)m$, answer length $\Lambda(m, \lfloor d/k \rfloor)(k-1)^{\lfloor d/k \rfloor}$, and probe complexity $\Lambda(d, \lfloor d/k \rfloor)$. Furthermore, if d is odd and $\lfloor d/k \rfloor = 1$ then the probe complexity can be reduced to d.*

Proof: Same as the proof of Theorem 6.1, where $\mathbf{P1}$ is replaced by $\mathbf{P1}'$ (see Lemma 5.4) or by $\mathbf{P1}''$ (see Lemma 5.5). □

As special cases, we get the following probe-efficient variants of Corollary 6.2 and Corollary 6.3.

Corollary 6.6 *For any positive integer k there exist:*

1. *A 1-private k-server protocol with communication complexity $O(k^3\, n^{1/(2k-1)})$ and probe complexity $2k-1$.*
2. *A 1-private 2-server protocol with communication complexity $4(6n)^{1/3} + 2 \approx 7.27n^{1/3}$ and probe complexity 3.*
3. *A 1-private k-server protocol with query length $O(\log n)$, answer length $O(n^{1/k+\epsilon})$, and probe complexity $O(n^{\epsilon})$, for any constant $\epsilon > 0$. More precisely, for any $\theta > 0$ there is a protocol with query length $(k-1)(1/H(\theta) + o(1)) \log n$, answer length $n^{H(\theta/k)/H(\theta)+o(1)}$, and probe complexity $n^{H(1/k)\theta/H(\theta)+o(1)}$.*

Proof: Similarly to the proofs of Corollary 6.2 and Corollary 6.3, the first bound is obtained from Theorem 6.5 by letting $d = 2k-1$ and $m = O(dn^{1/d})$ (in which case d is odd and $\lfloor d/k \rfloor = 1$), the second by letting $m = (6n)^{1/3}$ and $d = 3$, and the third by letting $m = (1/H(\theta) + o(1)) \log n$ and $d = \lfloor \theta m \rfloor$. □

Remark 6.7 In the protocols of Corollary 6.6 the user computes x_i by taking the exclusive-or of a subset of the answer bits, whose identity only depends on i. Moreover, the j-th element of this subset can be computed from the description of i in $O(\log n)$ space. Consequently, these protocols can be executed even by a log-space bounded user who cannot afford to store the queries or the random input r that was used to generate them.

6.2 Binary PIR Family

We obtain binary PIR protocols, in which each server sends a one-bit answer, by using $\mathbf{V2}$, $\mathbf{E2}$, $\mathbf{Shamir}$, and $\mathbf{P2}$.

Theorem 6.8 (Implicit in [18]) *Let m and d be positive integers such that $\binom{m+d}{d} \geq n$. Then, for any $t \geq 1$, there exists a t-private PIR protocol with $k = dt+1$ servers, query length $\lceil \log(k+1) \rceil m$, and answer length 1.*

Proof: Such a PIR protocol is obtained by letting $E = \mathbf{E2}$, $V = \mathbf{V2}_{d,m}$, $L = \mathbf{Shamir}_{k,t}$, and $P = \mathbf{P2}$, where F is GF($2^{\lceil \log(k+1) \rceil}$) – the smallest extension of GF(2) with at least $k+1$ elements, and the subfield F' used by $\mathbf{P2}$ is GF(2). □

Corollary 6.9 *For any $d, t \geq 1$ there is a t-private PIR protocol with $k = dt + 1$ servers, query length $O(d \log kn^{1/d})$, and answer length 1.*

Rephrasing the above corollary we get,

Corollary 6.10 *For any $k, t \geq 1$ there is a t-private k-server PIR protocol with query length*

$$O(\frac{k}{t} \log kn^{1/\lfloor (k-1)/t \rfloor}),$$

and answer length 1.

6.3 Probe-Efficient PIR Family

Our last family of protocols relies on $\mathbf{V3}$, $\mathbf{E3}$, $\mathbf{CNF}$, and $\mathbf{P3}$ as building blocks. These protocols have the interpretation of utilizing the "combinatorial cubes" geometry which was first used in [15]. In fact, they strictly generalize the 2-server protocol from [15]. The main advantage of the current family over the main family is that it provides probe-efficient protocols even in the case $t > 1$. It also slightly improves the probe complexity in the case $t = 1$. However, the communication complexity of protocols from this family is inferior to that of the main family, especially when $t > 1$.

We start with the case $t = 1$.

Theorem 6.11 *Let d and ℓ be positive integers such that $\ell^d \geq n$. Then, for any $k \geq 2$ there exists a 1-private k-server PIR protocol with query length $(k-1)d\ell$, answer length $\ell^{\lfloor d/k \rfloor}\binom{d}{\lfloor d/k \rfloor}$, and probe complexity $\binom{d}{\lfloor d/k \rfloor}$.*

Proof: The condition $\ell^d \geq n$ guarantees the existence of an encoding $\mathbf{E3}$ as required. We use the 1-private CNF secret-sharing scheme, thus the query length is $(k-1)d\ell$. Finally, we use $\mathbf{P3}$ as the SM protocol. Therefore, by Lemma 5.7, the answer length and the probe complexity are as promised. □

For general values of t, we get the following protocols.

Theorem 6.12 *Let d, ℓ, t be positive integers such that $\ell^d \geq n$. Then, for*

$$k = \min\left(\lfloor dt - (d + t - 3)/2 \rfloor \ , \ dt - t + 1 - (d \bmod 2)\right)$$

there exists a t-private k-server PIR protocol with query length $\binom{k-1}{t}d\ell$, answer length $d\ell$, and probe complexity d.

Proof: The condition $\ell^d \geq n$ guarantees the existence of the encoding $\mathbf{E3}$. The specified query length is induced by the use of the t-private CNF secret-sharing scheme. Finally, by Corollary 5.25, the answer length and the probe complexity are as promised. □

As an immediate corollary of Theorem 6.12 we get:

Corollary 6.13 *Fix any positive integers d, t. For $k = \min\left(\lfloor dt - (d + t - 3)/2 \rfloor \ , \ dt - t + 1 - (d \bmod 2)\right)$ there exists a t-private k-server PIR protocol with communication complexity $O_{d,t}(n^{1/d})$ and probe complexity d.*

The smallest interesting instance of Corollary 6.13 is a 2-private 4-server protocol with communication complexity $O(n^{1/3})$ and probe complexity 3. In comparison to the corresponding protocol of the main family (see Theorem 6.1), the communication complexity of this protocol is worse by a small constant factor, but its probe complexity is much better.

7 Optimized CNF Secret-Sharing

The families of PIR protocols, described so far, use two secret-sharing schemes: Shamir's scheme and the CNF scheme. Shamir's scheme has the smallest shares possible – the size of each share is the maximum between the secret size and the logarithm of the number of players [27]. In contrast, the size of each share in the CNF scheme is $\binom{k-1}{t}$ times the size of the secret. In some sense, the CNF sharing gives more redundancy to the share-holders. Our main family of protocols (described in Section 6.1) and the protocols with small probe complexity (described in Section 6.3) exploit this redundancy to improve the communication complexity compared to the protocols of [15]. This raises the question whether we can maintain the improved communication complexity without paying the penalty of the redundancy, i.e., with shorter queries. This penalty can be quite big in the t-private protocols.

We demonstrate that some savings are possible. Specifically, we construct a secret-sharing scheme $\mathbf{OptCNF}$, whose share complexity improves on that of the CNF scheme by roughly a factor of $t+1$ when $k \gg t$; yet, an SM protocol with identical communication complexity to that of $\mathbf{P1}$ (and significantly better time complexity) can be based on $\mathbf{OptCNF}$. This results in a similar improvement to the query length of our main family. An additional feature of the optimized construction is a significant reduction in the *computation* required by the servers. For instance, in the 1-private case its dependence on k is reduced from k^{2k-1} to roughly $k!$. We start with 1-private protocols and later discuss the t private case.

Definition 7.1 (1-Private optimized CNF scheme) To 1-privately share a secret $s \in F$:

- *Additively* share s into k pieces $r_1, \ldots, r_k$; that is, $s = \sum_{i=1}^{k} r_i$, where the pieces r_i are otherwise-random field elements.
- Define $z_i = \sum_{j=i+1}^{k} r_j$.
- Distribute to each player P_j the pieces $r_1, \ldots, r_{j-1}, z_j$.

The 1-privacy of the above scheme follows from the fact that each share contains no more information than the share of the same party in the secret-sharing scheme $\mathbf{CNF}_{k,1}$. On the other hand, every pair of parties, say P_{j_1}, P_{j_2} where $j_1 < j_2$, can reconstruct the secret s by computing $\sum_{j=1}^{j_1} r_j + z_{j_1}$ (all the r_j's in the sum, in particular r_{j_1}, are held by P_{j_2}). The total share size summed over all parties in this scheme is $(k+2)(k-1)/2$ field elements. Asymptotically, this improves the total share size by a factor of 2 with respect to $\mathbf{CNF}_{k,1}$, where the total share size is $k(k-1)$ field elements.

Lemma 7.2 *For $V = \mathbf{V1}_{d,m}$ and the 1-private optimized CNF sharing $\mathbf{OptCNF}$, there exists an SM protocol $\mathbf{P4}$ with message length $\Lambda(m, \lfloor d/k \rfloor)$.*

Proof: We present an SM protocol $\mathbf{P4}$ as required. The description uses the notation of Protocol $\mathbf{P1}$ presented in the proof of Lemma 5.1; we consider only the case where $t = 1$, and denote $Y_{\{j\},b}$ by $Y_{j,b}$. In Protocol $\mathbf{P1}$ the servers hold an m-variate polynomial p, which defines a km-variate polynomial q. The monomials of q are partitioned to k polynomials $q_1, \ldots, q_k$ such that q_j contains only monomials in which the number of the variables $Y_{j,1}, \ldots, Y_{j,m}$ in each monomial is at most $\lfloor d/k \rfloor$. As discussed in Remark 5.2, in $\mathbf{P1}$ we did not specify the exact partition, that is, how to assign monomials that could be assigned to more than one polynomial q_j. For our next construction it is essential to require that a monomial is assigned to the polynomial q_j, where j is the *smallest* index such that the number of the variables $Y_{j,1}, \ldots, Y_{j,m}$ in the monomial is at most $\lfloor d/k \rfloor$.

Fix any j and consider the server $\mathcal{S}_j$. In Protocol $\mathbf{P1}$ server $\mathcal{S}_j$ substitutes the values of the variables that it knows in q_j to obtain the polynomial $\hat{q}_j \stackrel{\text{def}}{=} q(y_{1,1}, \ldots, y_{j-1,m}, Y_{j,1}, \ldots Y_{j,m}, y_{j+1,1}, \ldots, y_{k,m})$. We claim that the new shares of $\mathbf{OptCNF}$ suffice for $\mathcal{S}_j$ to compute his original answer. Recall that every monomial in q is multi-linear, and furthermore, if a variable $Y_{j,\ell}$ appears in some monomial of q then for every $j' \neq j$ the variable $Y_{j',\ell}$ does not appear in that monomial. We define an equivalence relation between multi-linear monomials over the variables $Y_{1,1}, \ldots, Y_{1,m}, \ldots, Y_{k,1}, \ldots, Y_{k,m}$. (For every j we define a different equivalence relation.) We say that two monomials M_1 and M_2 are in the relation if:

1. For every $\ell \in [m]$ and $h \in \{1, \ldots, j\}$ the variable $Y_{h,\ell}$ appears in M_1 if and only if it appears in M_2, and
2. For every $\ell \in [m]$, there is some index $h_1 \in \{j+1, \ldots, k\}$ such that the variable $Y_{h_1,\ell}$ appears in M_1 if and only if there is some $h_2 \in \{j+1, \ldots, k\}$ such that the variable $Y_{h_2,\ell}$ appears in M_2, and
3. For every $h \in \{1, \ldots, j-1\}$ the monomial M_1 contains more than $\lfloor d/k \rfloor$ variables from the set $Y_{h,1}, \ldots, Y_{h,m}$. (By Item (1) this is also true for M_2.)

Also, if the condition in Item (3) does not hold for two monomials then they are in the relation. Notice that if a monomial M is assigned to q_j then all the monomials in its equivalent class appear in q and are assigned to q_j. By the multi-linearity, the sum of the monomials in each equivalence class can be expressed as a new monomial is the variables $Y_{1,1}, \ldots, Y_{1,m}, \ldots, Y_{j,1}, \ldots, Y_{j,m}$ and new variables $Z_{j,1}, \ldots, Z_{j,m}$ where $Z_{j,\ell} \stackrel{\text{def}}{=} \sum_{h=j+1}^{k} Y_{h,\ell}$. Furthermore, the polynomial q_j is the sum of the new monomials. By the definition of $\mathbf{OptCNF}$, server $\mathcal{S}_j$ knows the values $z_{j,1}, \ldots, z_{j,m}$ to be assigned to $Z_{j,1}, \ldots, Z_{j,m}$. Thus, $\mathcal{S}_j$ can compute the coefficients of $\hat{q}_j$ and send them to the user as in Protocol $\mathbf{P1}$. $\square$

We can generalize the optimized CNF scheme for arbitrary privacy thresholds. We only describe the secret-sharing scheme; the details of the appropriate SM protocol are the same as in the above described Protocol $\mathbf{P4}$.

Definition 7.3 (The t-private optimized CNF scheme) To t-privately share a secret $s \in F$:

- *Additively* share s into $\binom{k}{t}$ pieces, each labeled by a different set from $\binom{[k]}{t}$; that is, $s = \sum_{T \in \binom{[k]}{t}} r_T$, where the pieces r_T are otherwise-random field elements.
- For every $j \in [k]$ and every $A \subseteq [j-1]$ such that $|A| \leq t$ and $|A| \geq t+j-k$ define

$$z_{j,A} = \sum_{T: j \notin T,\ T \cap [j-1] = A} r_T.$$

- Distribute to each player P_j all pieces $z_{j,A}$ such that $A \subseteq [j-1]$ such that $|A| \leq t$ and $|A| \geq t+j-k$.

The t-privacy of the above scheme follows from the t-privacy of the basic $\mathbf{CNF}_{k,t}$ scheme. (While not necessary for our purposes, it can also be verified that any set $C \subseteq [k]$ such that $|C| > t$ can reconstruct the secret.) When $k \gg t$, this scheme improves the total share size by a factor of $t+1$ compared to $\mathbf{CNF}$.

Acknowledgments. We thank Tal Malkin, Mike Saks, Yoav Stahl, and Xiaodong Sun for helpful related discussions.

References

[1] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. of the 24th International Colloquium on Automata, Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407. Springer, 1997.

[2] A. Ambainis and S. Lokam. Improved upper bounds on the simultaneous messages complexity of the generalized addressing function. In *LATIN 2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 207 – 216. Springer, 2000.

[3] L. Babai, A. Gál, P. G. Kimmel, and S. V. Lokam. Communication complexity of simultaneous messages. *SIAM J. on Computing*, 33(1):137 – 166, 2003.

[4] L. Babai, P. G. Kimmel, and S. V. Lokam. Simultaneous messages vs. communication. In *12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 361–372. Springer, 1995. Journal version in [3].

[5] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In C. Choffrut and T. Lengauer, editors, *STACS '90, 7th Annu. Symp. on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer-Verlag, 1990.

[6] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *J. of Cryptology*, 10(1):17–36, 1997. Early version: Security with small communication overhead, CRYPTO '90, volume 537 of *Lecture Notes in Computer Science*, pages 62-76. Springer-Verlag, 1991.

[7] R. Beigel, L. Fortnow, and W. Gasarch. Nearly tight bounds for private information retrieval systems. Technical Report 2002-L001N, NEC Laboratories America, 2002.

[8] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In P. G. Spirakis and J. van Leeuven, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer, 2001.

[9] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the $O(n^{\frac{1}{2k-1}})$ barrier for information-theoretic private information retrieval. In *Proc. of the 43rd Annu. IEEE Symp. on Foundations of Computer Science*, pages 261–270, 2002.

[10] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J. of Cryptology*, 2004. To appear. Preliminary version: M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 56–74. Springer, 2000.

[11] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer-Verlag, 1990.

[12] D. Bleichenbacher, A. Kiayias, and M. Yung. Decoding of interleaved Reed Solomon codes over noisy data. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Proc. of the 30th International Colloquium on Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2003.

[13] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[14] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of the 29th Annu. ACM Symp. on the Theory of Computing*, pages 304–313, 1997.

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

[16] D. Coppersmith and M. Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proc. of the 35th Annu. ACM Symp. on the Theory of Computing*, pages 136–142, 2003.

[17] A. Deshpande, R. Jain, T Kavita, V. Lokam, and J. Radhakrishnan. Better lower bounds for locally decodable codes. In *Proc. of the 17th Annu. IEEE Conf. on Computational Complexity*, pages 184–193, 2002.

[18] G. Di-Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *J. of Cryptology*, 14(1):37–74, 2001. Preliminary version in *Proc. of the 17th Annu. ACM Symp. on Principles of Distributed Computing*, pages 91–100, 1998.

[19] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. of the 30th Annu. ACM Symp. on the Theory of Computing*, pages 151–160, 1998. Journal version: *J. of Computer and System Sciences*, 60(3):592–629, 2000.

[20] N. Gilboa and Y. Ishai. Compressing cryptographic resources. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO '99*, Lecture Notes in Computer Science, pages 591–608. Springer-Verlag, 1999.

[21] O. Goldreich, H. Karloff, L. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and PIR. In *Proc. of the 17th Annu. IEEE Conf. on Computational Complexity*, pages 175–183, 2002.

[22] Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. of the 31st Annu. ACM Symp. on the Theory of Computing*, pages 79 – 88, 1999.

[23] Y. Ishai and X. Sun. Personal communication, 2000.

[24] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structure. In *Proc. of the IEEE Global Telecommunication Conf., Globecom 87*, pages 99–102, 1987. Journal version: Multiple Assignment Scheme for Sharing Secret. *J. of Cryptology*, 6(1):15-20, 1993.

[25] T. Itoh. Efficient private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):11–20, 1999.

[26] T. Itoh. On lower bounds for the communication complexity of private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(1):157–164, 2001.

[27] E. D. Karnin, J. W. Greene, and M. E. Hellman. On secret sharing systems. *IEEE Trans. on Information Theory*, 29(1):35–41, 1983.

[28] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proc. of the 32nd Annu. ACM Symp. on the Theory of Computing*, pages 80–86, 2000.

[29] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Proc. of the 35th Annu. ACM Symp. on the Theory of Computing*, pages 106–115, 2003.

[30] A. Kiayias and M. Yung. Secure games with polynomial expressions. In P. G. Spirakis and J. van Leeuven, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 939–950. Springer, 2001.

[31] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science*, pages 364–373, 1997.

[32] J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1982.

[33] E. Mann. Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology, Haifa, 1998.

[34] K. Obata. Optimal lower bounds for 2-query locally decodable linear codes. In *RANDOM '02, 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 2483 of *Lecture Notes in Computer Science*, pages 39 – 50. Springer-Verlag, 2002.

[35] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[36] J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1998.

A Approximations of $\Lambda(m, d)$

For various calculations, we need to give bounds on $\Lambda(m, d)$. For an integer d, we can use the following approximation:

$$\Lambda(m, d) > \frac{(m-d)^d}{d!} \tag{2}$$

In particular, for $\Lambda(m, d) \geq n$ to hold, it is sufficient to let $m = (d!n)^{1/d} + d = \alpha_d d n^{1/d} + d$, where α_d is a constant depending on d. It holds that $\alpha_3 = (6)^{1/3}/3 \approx 0.61$, $\alpha_5 = (120)^{1/5}/5 \approx 0.52$, and $\alpha_d < 0.5$ for $d \geq 7$ (by the Stirling approximation $\lim_{d\to\infty} \alpha_d = 1/e$).

If $d = \lfloor \theta m \rfloor$ for some constant $\theta \leq 0.5$ we will use the following approximation.

$$2^{(H(\theta)-o(1))m} \leq \Lambda(m, \lfloor \theta m \rfloor) \leq 2^{H(\theta)m} \tag{3}$$

(cf. [32, Theorem 1.4.5]). In particular, for $\Lambda(m, \lfloor \theta m \rfloor) \geq n$ to hold, it is sufficient to let $m = (1/H(\theta) + o(1)) \log n$.

B Low-Degree Encoding

In this section we prove the validity of the low-degree encodings $\mathbf{E1}$ and $\mathbf{E2}$ defined in Section 4.1. For each of these encodings we need to show the existence of degree-d polynomials p_i such that $p_i(E(j))$ is 1 if $i = j$ and is zero otherwise.

Claim B.1 *The encoding* $\mathbf{E1}$ *is valid.*

Proof: Recall the definition of $\mathbf{E1}$. For n distinct vectors $v^1, \ldots, v^n$ in GF$(2)^m$ with Hamming weight at most d, we define $\mathbf{E1}(i) = v^i$. Assume, without loss of generality, that the weights of these vectors is monotonic non-increasing; i.e., if $j < i$ then the weight of v^j is greater or equal to the weight of v^i. Denote by S_i the subset of $[m]$ containing the positions in which v^i is 1. We define the polynomials p_i one after the other, starting with p_1 and ending at p_n: Let

$$p_i(Y_1, \ldots, Y_m) \stackrel{\text{def}}{=} \prod_{h \in S_i} Y_h - \sum_{j : S_i \subset S_j} p_j(Y_1, \ldots, Y_m). \tag{4}$$

If $S_i \subset S_j$ then the weight of v^j is greater than the weight of v^i and thus all the polynomials in the right hand side of (4) are already defined. Clearly, the degree of the polynomials p_i is at most d. We prove by induction on i that $p_i(v^j)$ equals 1 iff $i = j$. First, by the induction hypothesis, $p_i(v^i) = \prod_{h \in S_i} v^i_h - \sum_{j:S_i \subset S_j} p_j(v^i) = 1 - 0 = 1$. Second, if $j \neq i$ then $p_i(v^j) = \prod_{h \in S_i} v^j_h$ if $S_i \not\subset S_j$ and $p_i(v^j) = \prod_{h \in S_i} v^j_h - p_j(v^j)$ otherwise (again, by the induction hypothesis). In both cases, $p_i(v^j) = 0$. □

The proof of the validity of $\mathbf{E2}$ for fields with a prime number of elements can be found in [18, Lemma 6]. We (slightly) generalize it to any field.

Claim B.2 *The encoding* $\mathbf{E2}$ *is valid.*

Proof: Recall the definition of $\mathbf{E2}$. Let $\omega_0, \ldots, \omega_d$ be distinct elements from some field F. Then, $\mathbf{E2}(i)$ is the i-th vector of the form $(\omega_{f^i_1}, \ldots, \omega_{f^i_m})$ such that $\sum_{\ell=1}^m f^i_\ell \leq d$. First, define the following polynomials r_i (the definition of the polynomials p_i uses these polynomials).

$$r_i(Y_1, \ldots, Y_m) \stackrel{\text{def}}{=} \prod_{\ell=1}^{m} \prod_{h=0}^{f^i_\ell - 1} \frac{Y_\ell - \omega_h}{\omega_{f^i_\ell} - \omega_h}. \tag{5}$$

The degree of r_i is $\sum_{\ell=1}^m f^i_\ell \leq d$.

If $\sum_{\ell=1}^m f^i_\ell = d$ then we define $p_i \stackrel{\text{def}}{=} r_i$. Indeed, $r_i(\mathbf{E2}(i)) = 1$. On the other hand, for every $j \neq i$ it holds that $\sum_{\ell=1}^m f^j_\ell \leq d = \sum_{\ell=1}^m f^i_\ell$. Thus, there is at least one ℓ such that $f^j_\ell < f^i_\ell$ and $r_i(\mathbf{E2}(j)) = 0$.

For the general case, where $\sum_{\ell=1}^m f^i_\ell \leq d$, assume, without loss of generality, that if $j < i$, then $\sum_{\ell=1}^m f^i_\ell \leq \sum_{\ell=1}^m f^j_\ell$. Let

$$p_i(Y_1, \ldots, Y_m) \stackrel{\text{def}}{=} r_i(Y_1, \ldots, Y_m) - \sum_{j=1}^{i-1} p_j(Y_1, \ldots, Y_m) r_i(\mathbf{E2}(j)). \tag{6}$$

Clearly, the degree of the polynomials p_i is at most degree d. By induction on i and the arguments for the case $\sum_{\ell=1}^m f^i_\ell = d$ it follows that that $p_i(v^j)$ equals 1 iff $i = j$. □

C A Lower Bound on k_s

In this section we present a *tight* lower bound on $k_s(d,t)$ for the case $d = 3$. The most significant implication of this bound is a strong separation between the cover bound k_c and the span bound k_s over *any* field F.

Claim C.1 *For any $t > 1$, and over any field F,*

$$k_s(3,t) \geq 2t.$$

Proof: We show that with $k = 2t - 1$ the sum of all terms is not spanned. For this we define a set of terms and show that every linear combination of 0-blocks and 1-blocks misses at least one term.

First we define a "cycle" of sets

$$A - B_0 - C_0 - B_1 - C_1 - \ldots - B_{t-2} - C_{t-2} - A.$$

This is a cycle of odd length $(2t-1)$, each edge of which represents a pair of sets which we would like to cover $[k]$. Define the sets in this cycle as follows: each set will be taken to include t (cyclically) consecutive elements, where the "first" element in each set is the last element in the previous one. (Notice that the number of sets in the cycle suffices exactly for the first set to be consecutive to the last one.) For instance, for $t = 5$ and $k = 2t - 1 = 9$ the sets will be: $A = \{1,2,3,4,5\}$, $B_0 = \{5,6,7,8,9\}$, $C_0 = \{9,1,2,3,4\}$, $B_1 = \{4,5,6,7,8\}$, $C_1 = \{8,9,1,2,3\}$, $B_2 = \{3,4,5,6,7\}$, $C_2 = \{7,8,9,1,2\}$, $B_3 = \{2,3,4,5,6\}$, $C_3 = \{6,7,8,9,1\}$. Next, we construct the following list of $2t-3$ terms:

$$\begin{gathered}(A, B_0, C_0)\\(A, B_1, C_0)\\(A, B_1, C_1)\\(A, B_2, C_1)\\(A, B_2, C_2)\\\vdots\\(A, B_{t-2}, C_{t-3})\\(A, B_{t-2}, C_{t-2})\end{gathered}$$

From the way that the sets and the terms were constructed it follows that:

- None of the terms in the above list is a 0-block (since $B_j \cup C_j = B_j \cup C_{j-1} = [k]$).
- Any 1-block either covers no term, or covers two consecutive terms in the list (since there is no 1-block of the form $(*, B, C)$ that covers any of the above terms and each 1-block of the form $(A, B_j, *)$ or $(A, *, C_j)$ covers two terms).[15]

Now, assign to the terms with an odd position in the list the weight 1, to the terms with an even position in the list weight -1, and to all remaining terms in the term space the weight 0. It may be concluded from the above that each block has weight 0, and hence each linear combination of blocks has weight 0 as well. Since the sum of the weights of all terms is nonzero (as the list is of odd length), the sum of all terms cannot be spanned. □

We note that the lower bound of Claim C.1 is tight, since it matches the upper bound of Lemma 5.20. Finally, combining the lower bound on k_s provided by Claim C.1 with the exact characterization of k_c in Claim 5.15, we obtain the desired separation of k_s from k_c.

[15] Note that $(A, B_0, *)$ and $(A, *, C_{t-2})$ are *not* a valid 1-block.