On Computational Shortcuts for Information-Theoretic PIR

Matthew M. Hong[1⋆], Yuval Ishai[2⋆⋆], Victor I. Kolobov[2], and Russell W. F. Lai[3⋆⋆⋆]

[1] IIIS, Tsinghua University, Beijing, China
hoou8547@hotmail.com
[2] Technion, Haifa, Israel
{yuvali,tkolobov}@cs.technion.ac.il
[3] Friedrich-Alexander University Erlangen-Nuremberg, Nuremberg, Germany
russell.lai@cs.fau.de

Abstract. Information-theoretic *private information retrieval* (PIR) schemes have attractive concrete efficiency features. However, in the standard PIR model, the computational complexity of the servers must scale linearly with the database size.
We study the possibility of bypassing this limitation in the case where the database is a truth table of a "simple" function, such as a union of (multi-dimensional) intervals or convex shapes, a decision tree, or a DNF formula. This question is motivated by the goal of obtaining lightweight *homomorphic secret sharing* (HSS) schemes and secure multiparty computation (MPC) protocols for the corresponding families.
We obtain both positive and negative results. For "first-generation" PIR schemes based on Reed-Muller codes, we obtain computational shortcuts for the above function families, with the exception of DNF formulas for which we show a (conditional) hardness result. For "third-generation" PIR schemes based on matching vectors, we obtain stronger hardness results that apply to all of the above families. Our positive results yield new information-theoretic HSS schemes and MPC protocols with attractive efficiency features for simple but useful function families. Our negative results establish new connections between information-theoretic cryptography and fine-grained complexity.

1 Introduction

Secure multiparty computation (MPC) [61,49,15,27] allows two or more parties to compute a function of their secret inputs while only revealing the output. Much of

⋆ Work done in part while visiting Technion.
⋆⋆ Supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India.
⋆⋆⋆ Work done in part while visiting Technion. Supported by the State of Bavaria at the Nuremberg Campus of Technology (NCT). NCT is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN).

the large body of research on MPC is focused on minimizing *communication complexity*, which often forms an efficiency bottleneck. In the setting of computational security, fully homomorphic encryption (FHE) essentially settles the main questions about *asymptotic* communication complexity of MPC [46,24,47,23]. However, the information-theoretic (IT) analog of the question, *i.e.*, how communication-efficient IT MPC protocols can be, remains wide open, with very limited negative results [45,53,38,37,2,35,5]. These imply superlinear lower bounds only when the number of parties grows with the total input length. Here we will mostly restrict our attention to the simple case of a constant number of parties with security against a single, *passively* corrupted, party.

On the upper bounds front, the communication complexity of classical IT MPC protocols from [15,27] scales linearly with the *circuit size* of the function f being computed. With few exceptions, the circuit size remains a barrier even today. One kind of exceptions includes functions f whose (probabilistic) degree is smaller than the number of parties [9,6]. Another exception includes protocols that have access to a trusted source of correlated randomness [53,36,32,20]. Finally, a very broad class of exceptions that applies in the standard model includes "complex" functions, whose circuit size is super-polynomial in the input length. For instance, the minimal circuit size of most Boolean functions $f : \{0,1\}^n \to \{0,1\}$ is $2^{\tilde{\Omega}(n)}$. However, *all* such functions admit a 3-party IT MPC protocol with only $2^{\tilde{O}(\sqrt{n})}$ bits of communication [43,10]. This means that for most functions, communication is super-polynomially smaller than the circuit size. Curiously, the *computational complexity* of such protocols is bigger than 2^n even if f has circuits of size $2^{o(n)}$. These kind of gaps between communication and computation will be in the center of the present work.

Beyond the theoretical interest in the asymptotic complexity of IT MPC protocols, they also have appealing *concrete efficiency* features. Indeed, typical implementations of IT MPC protocols in the honest-majority setting are faster by orders of magnitude than those of similar computationally secure protocols for the setting of dishonest majority.[4] Even when considering *communication* complexity alone, where powerful tools such as FHE asymptotically dominate existing IT MPC techniques, the latter can still have better *concrete* communication costs when the inputs are relatively short. These potential advantages of IT MPC techniques serve to further motivate this work.

1.1 Homomorphic Secret Sharing and Private Information Retrieval

We focus on low-communication MPC in a simple client-server setting, which is captured by the notion of *homomorphic secret sharing* (HSS) [16,18,21]. HSS can be viewed as a relaxation of FHE which, unlike FHE, exists in the IT setting. In an HSS scheme, a client shares a secret input $x \in \{0,1\}^n$ between k servers. The servers, given a function f from some family $\mathcal{F}$, can *locally* apply an evaluation

[4] It is often useful to combine an IT protocol with a lightweight use of symmetric cryptography in order to reduce communication costs (see, e.g.,[48,33,3]); we will use such a hybrid approach in the context of optimizing concrete efficiency.

function on their input shares, and send the resulting output shares to the client. Given the k output shares, the client should recover $f(x)$. In the process, the servers should learn nothing about x, as long as at most t of them collude.

As in the case of MPC, we assume by default that $t = 1$ and consider a constant number of servers $k \geq 2$. A crucial feature of HSS schemes is *compactness* of output shares, typically requiring their size to scale linearly with the output size of f and independently of the complexity of f. This makes HSS a good building block for low-communication MPC. Indeed, HSS schemes can be converted into MPC protocols with comparable efficiency by distributing the input generation and output reconstruction [18].

An important special case of HSS is (multi-server) *private information retrieval* (PIR) [29]. A PIR scheme allows a client to retrieve a single bit from an N-bit database, which is replicated among $k \geq 2$ servers, such that no server (more generally, no t servers) learns the identity of the retrieved bit. A PIR scheme with database size $N = 2^n$ can be seen as an HSS scheme for the family $\mathcal{F}$ of *all* functions $f : \{0,1\}^n \to \{0,1\}$.

PIR in the IT setting has been the subject of a large body of work; see [63] for a partial survey. Known IT PIR schemes can be roughly classified into three generations. The first-generation schemes, originating from the work of Chor et al. [29], are based on Reed-Muller codes. In these schemes the communication complexity is $N^{1/\Theta(k)}$. In the second-generation schemes [13], the exponent vanishes super-linearly with k, but is still constant for any fixed k. Finally, the third-generation schemes, originating the works of Yekhanin [62] and Efremenko [43], have sub-polynomial communication complexity of $N^{o(1)}$ with only $k = 3$ servers or even $k = 2$ servers [41]. (An advantage of the 3-server schemes is that the server answer size is constant.) These schemes are based on a nontrivial combinatorial object called a *matching vectors* (MV) family.

As noted above, a PIR scheme with database size $N = 2^n$ can be viewed as an HSS scheme for the family $\mathcal{F}$ of all functions f (in truth-table representation). Our work is motivated by the goal of extending this to more expressive (and succinct) function representations. While a lot of recent progress has been made on the computational variant of the problem for functions represented by circuits or branching programs [17,18,39,44,54,22], almost no progress has been made for IT HSS. Known constructions are limited to the following restricted types: (1) HSS for general truth tables, corresponding to PIR, and (2) HSS for low-degree polynomials, which follow from the multiplicative property of Shamir's secret-sharing scheme [57,15,27,34]. Almost nothing is known about the existence of non-trivial IT HSS schemes for other useful function families, which we aim to explore in this work.

1.2 HSS via Computational Shortcuts for PIR

Viewing PIR as HSS for truth tables, HSS schemes for more succinct function representations can be equivalently viewed as a computationally efficient PIR schemes for *structured* databases, which encode the truth tables of succinctly described functions. While PIR schemes for general databases require linear

computation in N [14], there are no apparent barriers that prevent *computational shortcuts* for *structured databases*. In this work we study the possibility of designing useful HSS schemes by applying such shortcuts to existing IT PIR schemes. Namely, by exploiting the structure of truth tables that encode simple functions, the hope is that the servers can answer PIR queries with $o(N)$ computation.

We focus on the two main families of IT PIR constructions: (1) first-generation "Reed-Muller based" schemes, or RM PIR for short; and (2) third-generation "matching-vector based" schemes, or MV PIR for short. RM PIR schemes are motivated by their simplicity and their good concrete communication complexity on small to medium size databases, whereas MV PIR schemes are motivated by their superior asymptotic efficiency. Another advantage of RM PIR schemes is that they naturally scale to bigger security thresholds $t > 1$, increasing the number of servers by roughly a factor of t but maintaining the per-server communication complexity. For MV PIR schemes, the comparable t-private variants require at least 2^t servers [7].

1.3 Our Contribution

We obtain the following main results. See Section 2 for a more detailed and more technical overview.

Positive results for RM PIR. We show that for some natural function families, such as unions of multi-dimensional intervals or other convex shapes (capturing, e.g., geographical databases), decision trees, and DNF formulas with disjoint terms, RM PIR schemes do admit computational shortcuts. In some of these cases the shortcut is essentially optimal, in the sense that the computational complexity of the servers is equal to the size of the PIR queries plus the size of the function representation (up to polylogarithmic factors). In terms of concrete efficiency, the resulting HSS schemes can in some cases be competitive with alternative techniques from the literature, including lightweight computational HSS schemes based on symmetric cryptography [19], even for large domain sizes such as $N = 2^{40}$. This may come at the cost of either using more servers ($k \geq 3$ or even $k \geq 4$, compared to $k = 2$ in [19]) or alternatively applying communication balancing techniques from [29,11,60] that are only efficient for short outputs.

Negative results for RM PIR. The above positive result may suggest that "simple" functions admit shortcuts. We show that this can only be true to a limited extent. Assuming the Strong Exponential Time Hypothesis (SETH) assumption [26], a conjecture commonly used in fine-grained complexity [59], we show that there is no computational shortcuts for general DNF formulas. More broadly, there are no shortcuts for function families that contain hard counting problems.

Negative results for MV PIR. Somewhat unexpectedly, for MV PIR schemes, the situation appears to be significantly worse. Here we can show conditional hardness results even for the *all-1 database*. Of course, one can trivially realize an HSS scheme for the constant function $f(x) = 1$. However, our results effectively

rule out obtaining efficient HSS for richer function families via the MV PIR route, even for the simple but useful families to which our positive results for RM PIR apply. This shows a qualitative separation between RM PIR and MV PIR.

Our negative results are obtained by exploiting a connection between shortcuts in MV PIR and counting problems in graphs that we prove to be ETH-hard. While this only rules out a specific type of HSS constructions, it can still be viewed as a necessary step towards broader impossibility results. For instance, proving that (computationally efficient) HSS for simple function families cannot have $N^{o(1)}$ share size *inevitably requires* proving computational hardness of the counting problems we study, simply because if these problems were easy then such HSS schemes would exist. We stress that good computational shortcuts for MV PIR schemes, matching our shortcuts for RM PIR schemes, is a desirable goal. From a theoretical perspective, they would give rise to better information-theoretic HSS schemes for natural function classes. From an applied perspective, they could give concretely efficient HSS schemes and secure computation protocols (for the same natural classes) that outperform all competing protocols on moderate-sized input domains. (See the full version for communication break-even points.) Unfortunately, our negative results give strong evidence that, contrary to prior expectations, such shortcuts for MV PIR do not exist.

Positive results for tensored and parallel MV PIR. Finally, we show how to bypass our negative result for MV PIR via a "tensoring" operator and parallel composition. The former allows us to obtain the same shortcuts we get for RM PIR while maintaining the low communication cost of MV PIR, but at the cost of increasing the number of servers. This is done by introducing an exploitable structure similar to that in RM PIR through an operation that we called tensoring. In fact, tensoring can be applied to any PIR schemes with certain natural structural properties to obtain new PIR with shortcuts. The parallel composition approach is restricted to specific function classes and has a significant concrete overhead. Applying either transformation to an MV PIR scheme yields schemes that no longer conform to the baseline template of MV PIR, and thus the previous negative result does not apply.

2 Overview of Results and Techniques

Recall that the main objective of this work is to study the possibility of obtaining non-trivial IT HSS schemes via computational shortcuts for IT PIR schemes. In this section we give a more detailed overview of our positive and negative results and the underlying techniques.

From here on, we let $N = 2^n$ be the size of the (possibly structured) database, which in our case will be a truth table encoding a function $f : \{0,1\}^n \to \{0,1\}$ represented by a bit-string $\hat{f}$ of length $\ell = |\hat{f}| \leq N$. We are mostly interested in the case where $\ell \ll N$. We will sometimes use ℓ to denote a natural size parameter which is upper bounded by $|\hat{f}|$. For instance, $\hat{f}$ can be a DNF formula with ℓ terms over n input variables. We denote by $\mathcal{F}$ the *function family* associating each $\hat{f}$ with a function f and a size parameter ℓ, where $\ell = |\hat{f}|$ by default.

For both HSS and PIR, we consider the following efficiency measures:

- Input share size $\alpha(N)$: Number of bits that the client sends to each server.
- Output share size $\beta(N)$: Number of bits that each server sends to the client.
- Evaluation time $\tau(N, \ell)$: Running time of server algorithm, mapping an input share in $\{0,1\}^{\alpha(N)}$ and function representation $\hat{f} \in \{0,1\}^{\ell}$ to output share in $\{0,1\}^{\beta(N)}$.

When considering PIR (rather than HSS) schemes, we may also refer to $\alpha(N)$ and $\beta(N)$ as *query size* and *answer size* respectively. The computational model we use for measuring the running time $\tau(N, \ell)$ is the standard RAM model by default; however, both our positive and negative results apply (up to polylogarithmic factors) also to other standard complexity measures, such as circuit size.

Any PIR scheme PIR can be viewed as an HSS scheme for a truth-table representation, where the PIR database is the truth-table $\hat{f}$ of f. For this representation, the corresponding evaluation time τ must grow linearly with N. If a more expressive function family $\mathcal{F}$ supports faster evaluation time, we say that PIR admits a computational shortcut for $\mathcal{F}$. It will be useful to classify computational shortcuts as *strong* or *weak*. A strong shortcut is one in which the evaluation time is optimal up to polylogarithmic factors, namely $\tau = \tilde{O}(\alpha+\beta+\ell)$. (Note that $\alpha + \beta + \ell$ is the total length of input and output.) Weak shortcuts have evaluation time of the form $\tau = O(\ell \cdot N^{\delta})$, for some constant $0 < \delta < 1$. A weak shortcut gives a meaningful speedup whenever $\ell = N^{o(1)}$.

2.1 Shortcuts in Reed-Muller PIR

The first generation of PIR schemes, originating from the work of Chor et al. [29], represent the database as a low-degree multivariate polynomial, which the servers evaluate on each of the client's queries. We refer to PIR schemes of this type as *Reed-Muller* PIR (or RM PIR for short) since the answers to all possible queries form a Reed-Muller encoding of the database. While there are several variations of RM PIR in the literature, the results we describe next are insensitive to the differences. In the following focus on a slight variation of the original k-server RM PIR scheme from [29] (see [11]) that has answer size $\beta = 1$, which we denote by $\mathsf{PIR}^k_{\mathsf{RM}}$. For the purpose of this section we will mainly focus on the computation performed by the servers, for the simplest case of $k = 3$ ($\mathsf{PIR}^3_{\mathsf{RM}}$), as this is the aspect we aim to optimize. For a full description of the more general case we refer the reader to Section 4.

Let $\mathbb{F} = \mathbb{F}_4$ be the Galois field of size 4. In the $\mathsf{PIR}^3_{\mathsf{RM}}$ scheme, the client views its input $i \in [N]$ as a pair of indices $i = (i_1, i_2) \in [\sqrt{N}] \times [\sqrt{N}]$ and computes two vectors $q_1^j, q_2^j \in \mathbb{F}^{\sqrt{N}}$ for each server $j \in \{1,2,3\}$, such that $\{q_1^j\}$ depend on i_1 and $\{q_2^j\}$ depend on i_2. Note that this implies that $\alpha(N) = O(\sqrt{N})$. Next, each server j, which holds a description of a function $f\colon [\sqrt{N}] \times [\sqrt{N}] \to \{0,1\}$, computes an answer $a_j = \sum_{i'_1, i'_2 \in [\sqrt{N}]} f(i'_1, i'_2) q_1^j[i'_1] q_2^j[i'_2]$ with arithmetic over $\mathbb{F}$ and sends the client a single bit which depends on a_j (so $\beta(N) = 1$). The client reconstructs $f(i_1, i_2)$ by taking the exclusive-or of the 3 answer bits.

Positive Results for RM PIR The computation of each server j, $a_j = \sum_{i'_1,i'_2\in[\sqrt{N}]} f(i'_1,i'_2)q_1^j[i'_1]q_2^j[i'_2]$, can be viewed as an evaluation of a multivariate degree-2 polynomial, where $\{f(i'_1,i'_1)\}$ are the coefficients, and the entries of q_1^j, q_2^j are the variables. Therefore, to obtain a computational shortcut, one should look for *structured* polynomials that can be evaluated in time $o(N)$. A simple but useful observation is that computational shortcuts exist for functions f which are *combinatorial rectangles*, that is, $f(i_1,i_2)=1$ if and only if $i_1\in I_1$ and $i_2\in I_2$, where $I_1, I_2 \subseteq [\sqrt{N}]$. Indeed, we may write

$$a_j = \sum_{i'_1,i'_2\in[\sqrt{N}]} f(i'_1,i'_2)q_1^j[i'_1]q_2^j[i'_2] = \sum_{(i'_1,i'_2)\in(I_1,I_2)} q_1^j[i'_1]q_2^j[i'_2] \tag{1}$$

$$= \left(\sum_{i'_1\in I_1} q_1^j[i'_1]\right)\left(\sum_{i'_2\in I_2} q_2^j[i'_2]\right). \tag{2}$$

Note that if a server evaluates the expression using Equation (1) the time is $O(N)$, but if it instead uses Equation (2) the time is just $O(\sqrt{N}) = O(\alpha(N))$. Following this direction, we obtain non-trivial IT HSS schemes for some natural function classes such as disjoint unions of intervals and decision trees.

Theorem 1 (Decision trees, formal version Theorem 9). $\mathsf{PIR}^k_{\mathsf{RM}}$ *admits a weak shortcut for decision trees (more generally, disjoint DNF formulas). Concretely, for n variables and ℓ leaves (or terms), we have $\tau(N,\ell) = O(\ell \cdot N^{1/(k-1)})$, where $N = 2^n$.*

Theorem 2 (Union of disjoint intervals, formal version Theorems 10 and 11). *For every positive integers $d \geq 1$ and $k \geq 3$ such that $d|k-1$, $\mathsf{PIR}^k_{\mathsf{RM}}$ admits a strong shortcut for unions of ℓ disjoint d-dimensional intervals in $\left([N^{1/d}]\right)^d$. Concretely, $\tau(N,\ell) = O(N^{1/(k-1)} + \ell)$.*

Better shortcuts running in $\tilde{O}(N^{1/(k-1)} + \ell \cdot N^{1/3(k-1)})$ are also possible. Moreover, by expressing (discretized) convex bodies as unions of intervals, we generalize the result for interval functions to convex body membership functions.

Negative Results for RM PIR All of the previous positive results apply to function families $\mathcal{F}$ for which there is an efficient *counting algorithm* that given $\hat{f}\in\mathcal{F}$ returns the number of satisfying assignments of f. We show that this is not a coincidence: efficient counting can be reduced to finding a shortcut for $\hat{f}$ in $\mathsf{PIR}^k_{\mathsf{RM}}$. This implies that computational shortcuts are impossible for function representations for which the counting problem is hard. Concretely, following a similar idea from [52], we show that a careful choice of PIR query can be used to obtain the *parity* of all evaluations of f as the PIR answer. The latter is hard to compute even for DNF formulas, let alone stronger representation models, assuming standard conjectures from fine-grained complexity: either the *Strong Exponential Time Hypothesis* (SETH) or, with weaker parameters, even the standard *Exponential Time Hypothesis* (ETH) [26,25].

Theorem 3 (No shortcuts for DNF under ETH, formal version Corollaries 2 and 3). *Assuming (standard) ETH,* $\mathsf{PIR}^k_{\mathsf{RM}}$ *does not admit a strong shortcut for DNF formulas for sufficiently large* k*. Moreover, assuming SETH, for any* $k \geq 3$*,* $\mathsf{PIR}^k_{\mathsf{RM}}$ *does not admit a weak shortcut for DNF formulas.*

2.2 Hardness of Shortcuts for Matching-Vector PIR

Recall that MV PIR schemes are the only known PIR schemes achieving sub-polynomial communication (that is, $N^{o(1)}$) with a constant number of servers. We give strong evidence for hardness of computational shortcuts for MV PIR. We start with a brief technical overview of MV PIR.

We consider here a representative instance of MV PIR from [43,12], which we denote by $\mathsf{PIR}^3_{\mathsf{MV},\mathsf{SC}}$. This MV PIR scheme is based on two crucial combinatorial ingredients: a family of *matching vectors* and a *share conversion* scheme, respectively. We describe each of these ingredients separately.

A family of matching vectors MV consists of N pairs of vectors $\{u_x, v_x\}$ such that each matching inner product $\langle u_x, v_x\rangle$ is 0, and each non-matching inner product $\langle u_x, v_{x'}\rangle$ is nonzero. More precisely, such a family is parameterized by integers m, h, N and a subset $S \subset \mathbb{Z}_m$ such that $0 \notin S$. A matching vector family is defined by two sequences of N vectors $\{u_x\}_{x\in[N]}$ and $\{v_x\}_{x\in[N]}$, where $u_x, v_x \in \mathbb{Z}_m^h$, such that for all $x \in [N]$ we have $\langle u_x, v_x\rangle = 0$, and for all $x, x' \in [N]$ such that $x \neq x'$ we have $\langle u_x, v_{x'}\rangle \in S$. We refer to this as the *S-matching* requirement. Typical choices of parameters are $m = 6$ or $m = 511$ (products of two primes), $|S| = 3$ (taking the values $(0,1), (1,0), (1,1)$ in Chinese remainder notation), and $h = N^{o(1)}$ (corresponding to the PIR query length).

A share conversion scheme SC is a local mapping (without interaction) of shares of a secret y to shares of a related secret y', where $y \in \mathbb{Z}_m$ and y' is in some other Abelian group $\mathbb{G}$. Useful choices of $\mathbb{G}$ include $\mathbb{F}_2^2$ and $\mathbb{F}_2^9$ corresponding to $m = 6$ and $m = 511$ respectively. The shares of y and y' are distributed using linear secret-sharing schemes $\mathcal{L}$ and $\mathcal{L}'$ respectively, where $\mathcal{L}'$ is typically additive secret sharing over $\mathbb{G}$. The relation between y and y' that SC should comply with is defined by S as follows: if $y \in S$ then $y' = 0$ and if $y = 0$ then $y' \neq 0$. More concretely, if $(y_1, \ldots, y_k)$ are $\mathcal{L}$-shares of y, then each server j can run the share conversion scheme on (j, y_j) and obtain $y'_j = \mathsf{SC}(j, y_j)$ such that $(y'_1, \ldots, y'_k)$ are $\mathcal{L}'$-shares of some y' satisfying the above relation. What makes share conversion nontrivial is the requirement that the relation between y and y' hold regardless of the randomness used by $\mathcal{L}$ for sharing y.

Suppose MV and SC are compatible in the sense that they share the same set S. Moreover, suppose that SC applies to a 3-party linear secret-sharing scheme $\mathcal{L}$ over $\mathbb{Z}_m$. Then we can define a 3-server PIR scheme $\mathsf{PIR}^3_{\mathsf{MV},\mathsf{SC}}$ in the following natural way. Let $f : [N] \to \{0,1\}$ be the servers' database and $x \in [N]$ be the client's input. The queries are obtained by applying $\mathcal{L}$ to independently share each entry of u_x. Since $\mathcal{L}$ is linear, the servers can locally compute, for each $x' \in [N]$, $\mathcal{L}$-shares of $y_{x,x'} = \langle u_x, v_{x'}\rangle$. Note that $y_{x,x} = 0 \in \mathbb{Z}_m$ and $y_{x,x'} \in S$ (hence $y_{x,x'} \neq 0$) for $x \neq x'$. Letting $y_{j,x,x'}$ denote the share of $y_{x,x'}$ known to server j, each server can now apply share conversion to obtain a $\mathcal{L}'$-share

$y'_{j,x,x'} = \mathsf{SC}(j, y_{j,x,x'})$ of $y'_{x,x'}$, where $y'_{x,x'} = 0$ if $x \neq x'$ and $y'_{x,x'} \neq 0$ if $x = x'$. Finally, using the linearity of $\mathcal{L}'$, the servers can locally compute $\mathcal{L}'$-shares $\tilde{y}_j$ of $\tilde{y} = \sum_{x' \in [N]} f(x') \cdot y'_{x,x'}$, which they send as their answers to the client. Note that $\tilde{y} = 0$ if and only if $f(x) = 0$. Hence, the client can recover $f(x)$ by applying the reconstruction of $\mathcal{L}'$ to the answers and comparing $\tilde{y}$ to 0. When $\mathcal{L}'$ is additive over $\mathbb{G}$, each answer consists of a single element of $\mathbb{G}$.

Shortcuts for MV PIR Imply Subgraph Counting The question we ask in this work is whether the server computation in the above scheme can be sped up when f is a "simple" function, say one for which our positive results for RM PIR apply. Somewhat unexpectedly, we obtain strong evidence against this by establishing a connection between computational shortcuts for $\mathsf{PIR}^3_{\mathsf{MV},\mathsf{SC}}$ for useful choices of $(\mathsf{MV}, \mathsf{SC})$ and the problem of counting induced subgraphs. Concretely, computing a server's answer on the all-1 database and query x^j requires computing the parity of the number of subgraphs with certain properties in a graph defined by x^j. By applying results and techniques from parameterized complexity [28,42], we prove ETH-hardness of computational shortcuts for variants of the MV PIR schemes from [43,12]. In contrast to the case of RM PIR, these hardness results apply even for functions as simple as the constant function $f(x) = 1$.

The variants of MV PIR schemes to which our ETH-hardness results apply differ from the original PIR schemes from [43,12] only in the parameters of the matching vectors, which are worse asymptotically, but still achieve $N^{o(1)}$ communication complexity. The obstacle which prevents us from proving a similar hardness result for the original schemes from [43,12] seems to be an artifact of the proof, instead of an inherent limitation (more on this later). We therefore formulate a clean hardness-of-counting conjecture that would imply a similar hardness result for the original constructions from [43,12].

We now outline the ideas behind the negative results, deferring the technical details to Section 5. Recall that the computation of each server j in $\mathsf{PIR}^3_{\mathsf{MV},\mathsf{SC}}$ takes the form

$$\sum_{x' \in [N]} f(x') \cdot \mathsf{SC}(j, y_{j,x,x'}),$$

where $y_{j,x,x'}$ is the j-th share of $\langle u_x, v_{x'} \rangle$. Therefore, for the all-1 database ($f = 1$), for every S-matching vector family MV and share conversion scheme SC from $\mathcal{L}$ to $\mathcal{L}'$ we can define the $(\mathsf{MV}, \mathsf{SC})$-counting problem $\#(\mathsf{MV}, \mathsf{SC})$.

Definition 1 (Server computation problem). *For a Matching Vector family* MV *and share conversion* SC*, the problem* $\#(\mathsf{MV}, \mathsf{SC})$ *is defined as follows.*

- Input: *a valid* $\mathcal{L}$*-share* y_j *of some* $u_x \in \mathbb{Z}_m^h$ *(element-wise),*
- Output: $\sum_{x' \in [N]} \mathsf{SC}(j, y_{j,x,x'})$*, where* $y_{j,x,x'}$ *is the share of* $\langle u_x, v_{x'} \rangle$*.*

Essentially, the server computes N inner products of the input and the matching vectors using the homomorphic property of the linear sharing, maps the results using the share conversion and adds the result to obtain the final output.

Let $\mathsf{MV}^w_{\mathrm{Grol}}$ be a matching vectors family due to Grolmusz [50,40], which is used in all third-generation PIR schemes (see Section 5, Fact 1). For presentation, we focus on the special case $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}_{\mathrm{Efr}})$, where $\mathsf{SC}_{\mathrm{Efr}}$ is a share conversion due to Efremenko [43], which we present in Section 3.3. Note that all the results that follow also hold for the share conversion of [12], denoted by $\mathsf{SC}_{\mathrm{BIKO}}$. The family we consider, $\mathsf{MV}^w_{\mathrm{Grol}}$, is associated with the parameters $r \in \mathbb{N}$ and $w\colon \mathbb{N} \to \mathbb{N}$, such that the size of the matching vector family is $\binom{r}{w(r)}$, and the length of each vector is $h = \binom{r}{\leq \Theta\left(\sqrt{w(r)}\right)}$. By choosing $w(r) = \Theta(\sqrt{r})$ and r such that $N \leq \binom{r}{w(r)}$, the communication complexity of $\mathsf{PIR}^k_{\mathsf{MV}^w_{\mathrm{Grol}},\mathsf{SC}_{\mathrm{Efr}}}$ is $h = 2^{O(\sqrt{n \log n})}$, where $N = 2^n$, which is the best asymptotically among known PIR schemes.

Next, we relate $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}_{\mathrm{Efr}})$ to $\oplus\textsc{IndSub}(\Phi, w)$, the problem of deciding the parity of the number of w-node subgraphs of a graph G that satisfy graph property Φ. Here we consider the parameter w to be a function of the number of nodes of G. We will be specifically interested in graph properties $\Phi = \Phi_{m,\Delta}$ that include graphs whose number of edges modulo m is equal to Δ. Formally:

Definition 2 (Subgraph counting problem). *For a graph property Φ and parameter $w\colon \mathbb{N} \to \mathbb{N}$ (function of the number of nodes), the problem $\oplus\textsc{IndSub}(\Phi, w)$ is defined as follows.*

- Input: *Graph G with r nodes.*
- Output: *The parity of the number of induced subgraphs H of G such that: (1) H has $w(r)$ nodes; (2) $H \in \Phi$.*

We let $\Phi_{m,\Delta}$ denote the set of graphs H such that $|E(H)| \equiv \Delta \mod m$.

The following main technical lemma for this section relates obtaining computational shortcuts for $\mathsf{PIR}^k_{\mathsf{MV},\mathsf{SC}}$ to counting induced subgraphs.

Lemma 1 (From MV PIR to subgraph counting). *If $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}_{\mathrm{Efr}})$ can be computed in $N^{o(1)}$ $\left(= r^{o(w)}\right)$ time, then $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ can be decided in $r^{o(w)}$ time, for any nondecreasing function $w\colon \mathbb{N} \to \mathbb{N}$.*

The Hardness of Subgraph Counting The problem $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ is studied in parameterized complexity theory [42] and falls into the framework of *motif counting problems* described as follows in [56]: *Given a large structure and a small pattern called the motif, compute the number of occurrences of the motif in the structure.* In particular, the following result can be derived from Döfer *et al.* [42].

Theorem 4. *[42, Corollary of Theorem 22] $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ cannot be solved in time $r^{o(w)}$ unless* ETH *fails.*

Theorem 4 is insufficient for our purposes since it essentially states that no machine running in time $r^{o(w)}$ can successfully decide $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ for any pair (r, w). It other words, it implies hardness of counting for *some* weight parameter w, while for our case, we have specific function $w(r)$.

Fortunately, in [28] it was shown the counting of cliques, a very central motif, is hard for cliques of any size as long as it is bounded from above by $O(r^c)$ for an arbitrary constant $c < 1$ ($\sqrt{r}$, $\log r$, $\log^* r$, etc.), assuming ETH. Indeed, after borrowing results from [28] and via a more careful analysis of the proof of [42, Theorem 22], we can prove the following stronger statement about its hardness.

Theorem 5. *For some efficiently computable function $w = \Theta(\log r / \log\log r)$, $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ cannot be solved in time $r^{o(w)}$, unless ETH fails.*

Denote by MV^* the family $\mathsf{MV}^w_{\mathrm{Grol}}$ with $w(r) = \Theta(\log r / \log\log r)$ as in Theorem 5. Lemma 1 and Theorem 5 imply the impossibility result for strong shortcuts for PIR schemes instantiated with MV^*. Note that such an instantiation of $\mathsf{MV}^w_{\mathrm{Grol}}$ yields PIR schemes with subpolynomial communication $2^{O(n^{3/4}\mathrm{polylog}\, n)}$.

Theorem 6. *[No shortcuts in Efremenko MV PIR, formal version Theorem 15] $\#(\mathsf{MV}^*, \mathcal{SC}_{\mathrm{Efr}})$ cannot be computed in $N^{o(1)}$ $\left(= r^{o(w)}\right)$ time, unless ETH fails. Consequently, there are no strong shortcuts for the all-1 database for $\mathsf{PIR}^3_{\mathsf{MV}^*,\mathsf{SC}_{\mathrm{Efr}}}$.*

It is natural to ask whether hardness for other ranges of parameters such as $w = \Theta(\sqrt{r})$ holds for $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ in the spirit of Theorem 5. This is also of practical concern because the best known $\mathrm{MV}^w_{\mathrm{Grol}}$ constructions fall within such ranges. In particular, if we can show $\oplus\textsc{IndSub}(\Phi_{511,0}, \Theta(\sqrt{r}))$ cannot be decided in $r^{o(\sqrt{r})}$ time, it will imply that $\mathsf{PIR}^k_{\mathcal{P},\mathcal{C}}$ for $\mathcal{P} = \mathsf{MV}^{\Theta(\sqrt{r})}_{\mathrm{Grol}}$ and $\mathcal{C} = \mathsf{SC}_{\mathrm{Efr}}$ does not admit strong shortcuts for the all-1 database, since $\alpha(n) = N^{o(1)}$ but $\tau(n) = N^{\Omega(1)}$.

In fact, the problem $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ is plausibly hard, and can be viewed as a variant of the fine-grained-hard Exact-k-clique problem [59]. Consequently, we make the following conjecture.

Conjecture 1 (Hardness of counting induced subgraphs). $\oplus\textsc{IndSub}(\Phi_{m,\Delta}, w)$ cannot be decided in $r^{o(w)}$ time, for any integers $m \geq 2$, $0 \leq \Delta < m$, and for every function $w(r) = O(r^c)$, $0 \leq c < 1$.

For the impossibility results in this paper, we are only concerned with $w(r) = \Theta(\sqrt{r})$, and $(m, \Delta) = (511, 0)$ or $(m, \Delta) = (6, 4)$.

2.3 HSS from Generic Compositions of PIRs

Our central technique for obtaining shortcuts in PIR schemes is by exploiting the structure of the database. For certain PIR schemes where the structure is not exploitable, such as those based on matching vectors, we propose to introduce exploitable structures artificially by composing several PIR schemes. Concretely, we present two generic ways, tensoring and parallel PIR composition, to obtain a PIR which admits shortcuts for some function families by composing PIRs which satisfy certain natural properties. For details, we refer to the full version.

Tensoring First we define a *tensoring operation* on PIR schemes, which generically yields PIRs with shortcuts, at the price of increasing the number of servers.

Theorem 7 (Tensoring, informal). *Let* PIR *be a* k*-server PIR scheme satisfying some natural properties. Then there exists a* k^d*-server PIR scheme* $\mathsf{PIR}^{\otimes d}$ *with the same (per server) communication complexity that admits the same computational shortcuts as* $\mathsf{PIR}^{d+1}_{\mathsf{RM}}$ *does.*

When PIR is indeed instantiated with a matching-vector PIR, Theorem 7 gives HSS schemes for disjoint DNF formulas or decision trees with the best asymptotic efficiency out of the ones we considered.

Corollary 1 (Decision trees from tensoring, informal). *There is a* 3^d*-server HSS for decision trees, or generally disjoint DNF formulas, with* $\alpha(N) = \tilde{O}\left(2^{6\sqrt{n \log n}}\right)$, $\beta(N) = O(1)$ *and* $\tau(N, \ell) = \tilde{O}\left(N^{1/d+o(1)} + \ell \cdot N^{1/3d}\right)$, *where* n *is the number of variables and* ℓ *is the number of leaves in the decision tree.*

Parallel PIR Composition For the special case of interval functions, we can do even better with the second technique. We show that by making *parallel invocations* to HSS for point functions, it is possible to obtain HSS for the class of *sparsely-supported DNF functions*. In particular, this yields an HSS for union of intervals with the best asymptotic complexity among our constructions. This approach however does not generalize to better asymptotic results for decision trees or DNF formulas due to known lower bounds for covering codes [30].

Theorem 8 (Intervals from parallel composition, informal). *There is a* 3*-server HSS for unions of* ℓ d*-dimensional intervals with* $\alpha(N) = \tilde{O}\left(2^{6\sqrt{n \log n}}\right)$, $\beta(N) = O(\log(\frac{1}{\epsilon}))$ *and* $\tau(N, \ell) = \tilde{O}\left(\log(\frac{1}{\epsilon})\ell \cdot 2^{6\sqrt{n \log n}}\right)$.

2.4 Concrete Efficiency

Motivated by a variety of real-world applications, the concrete efficiency of PIR has been extensively studied in the applied cryptography and computer security communities; see, e.g., [31,51,55,58,1] and references therein. Many of the application scenarios of PIR can pontentially benefit from the more general HSS functionality we study in this work. To give a sense of the concrete efficiency benefits we can get, consider following MPC task: The client holds a secret input x and wishes to know if x falls in a union of a set of 2-dimensional intervals held by k servers, where at most t servers may collude ($t = 1$ by default). This can be generalized to return a payload associated with the interval to which x belongs. HSS for this "union of rectangles" function family can be useful for securely querying a geographical database.

We focus here on HSS obtained from the $\mathsf{PIR}^k_{\mathsf{RM}}$ scheme, which admits strong shortcuts for multi-dimensional intervals and at the same time offers attractive concrete communication complexity. For the database sizes we consider, the

concrete communication and computation costs are much better than those of (computational) single-server schemes based on fully homomorphic encryption. Classical secure computation techniques are not suitable at all for our purposes, since their communication cost would scale linearly with the number of intervals. The closest competing solutions are obtained via symmetric-key-based *function secret sharing* (FSS) schemes for intervals [17,19] (see full version for details).

We instantiate the FSS-based constructions with $k = 2$ servers, since the communication complexity in this case is only $O(\lambda n^2)$ for a security parameter λ [19]. For $k \geq 3$ (and $t = k - 1$), the best known FSS schemes require $O(\lambda\sqrt{N})$ communication [17]. Our comparison focuses on communication complexity which is easier to measure analytically. Our shortcuts make the computational cost scale linearly with the server input size, with small concrete constants. Below we give a few data points to compare the IT-PIR and the FSS-based approaches.

For a 2-dimensional database of size $2^{30} = 2^{15} \times 2^{15}$ (which is sufficient to encode a 300×300 km^2 area with 10×10 m^2 precision), the HSS based on $\mathsf{PIR}^k_{\mathsf{RM}}$ with shortcuts requires 16.1, 1.3, and 0.6 KB of communication for $k = 3, 4$ and 5 respectively, whereas FSS with $k = 2$ requires roughly 28 KB. For these parameters, we expect the concrete computational cost of the PIR-based HSS to be smaller as well.

We note that in $\mathsf{PIR}^k_{\mathsf{RM}}$ the payload size contributes additively to the communication complexity. If the payload size is small (a few bits), it might be beneficial to base the HSS on a "balanced" variant of $\mathsf{PIR}^k_{\mathsf{RM}}$ proposed by Woodruff and Yekhanin [60]. Using the Baur-Strassen algorithm [8], we can get the same shortcuts as for $\mathsf{PIR}^k_{\mathsf{RM}}$ with rougly half as many servers, at the cost of longer output shares that have comparable size to the input shares. Such balanced schemes are more attractive for short payloads than for long ones. For a 2-dimensional database of size $2^{30} = 2^{15} \times 2^{15}$, the HSS based on balanced $\mathsf{PIR}^k_{\mathsf{RM}}$ with 1-bit payload requires 1.5 and 0.2 KB communication for $k = 2$ and 3 respectively.

Our approach is even more competitive in the case of a higher corruption threshold $t \geq 2$, since (as discussed above) known FSS schemes perform more poorly in this setting, whereas the cost of $\mathsf{PIR}^k_{\mathsf{RM}}$ scales linearly with t. Finally, $\mathsf{PIR}^k_{\mathsf{RM}}$ is more "MPC-friendly" than the FSS-based alternative in the sense that its share generation is non-cryptographic and thus is easier to distribute via an MPC protocol.

3 Preliminaries

Let $m, n \in \mathbb{N}$ with $m \leq n$. We use $\{0,1\}^n$ to denote the set of bit strings of length n, $[n]$ to denote the set $\{1, \ldots, n\}$, and $[m, n]$ to denote the set $\{m, m+1, \ldots, n\}$. The set of all finite-length bit strings is denoted by $\{0,1\}^*$. Let $v = (v_1, \ldots, v_n)$ be a vector. We denote by $v[i]$ or v_i the i-th entry v. Let S, X be sets with $S \subseteq X$. The set membership indicator $\chi_{S,X} : X \to \{0,1\}$ is a function which outputs 1 on input $x \in S$, and outputs 0 otherwise. When X is clear from the context, we omit X from the subscript and simply write χ_S.

3.1 Function Families

To rigorously talk about a function and its description as separate objects, we define function families in a fashion similar to that in [17].

Definition 3 (Function Families). *A function family is a collection of tuples* $\mathcal{F} = \{\mathcal{F}_n = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E_n)\}_{n\in\mathbb{N}}$ *where* $\mathcal{X}_n \subseteq \{0,1\}^*$ *is a domain set,* $\mathcal{Y}_n \subseteq \{0,1\}^*$ *is a range set,* $P_n \subseteq \{0,1\}^*$ *is a collection of function descriptions, and* $E_n : P_n \times \mathcal{X}_n \to \mathcal{Y}_n$ *is an algorithm, running in time* $O(|\mathcal{X}_n|)$*, defining the function described by each* $\hat{f} \in P_n$*.*

Concretely, each $\hat{f} \in P_n$ *describes a corresponding function* $f : \mathcal{X}_n \to \mathcal{Y}_n$ *defined by* $f(x) = E_n(\hat{f}, x)$*. Unless specified, from now on we assume that* $\mathcal{X}_n = \{0,1\}^n$ *and* $\mathcal{Y}_n = \mathbb{F}_2$*. When there is no risk of confusion, we will describe a function family by* $\mathcal{F}_n$ *instead of* $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$*, write* f *instead of* $\hat{f}$*, and write* $f \in \mathcal{F}_n$ *or* $f \in \mathcal{F}$ *instead of* $\hat{f} \in P_n$*.*

Definition 4 (All Boolean Functions). *The family of all Boolean functions is a tuple* $\mathrm{ALL}_n = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E_n)$ *where* P_n *is a set containing the truth table* $\hat{f}$ *of* f *for each* $f : \mathcal{X}_n \to \mathcal{Y}_n$*, and* E_n *is the selection algorithm such that* $E_n(\hat{f}, x) = \hat{f}[x]$*.*

We next define combinatorial rectangle functions, each of which is parameterized with a combinatorial rectangle, and it outputs 1 whenever the input lies in the rectangle. This family is central to the shortcuts that we obtain for the Reed-Muller PIR and the PIRs obtained by tensoring.

Definition 5 (Combinatorial Rectangles). *Let* $d \in \mathbb{N}$*,* $\mathcal{X}^1, \ldots, \mathcal{X}^d$ *be sets and* $\mathsf{cr} : \mathcal{X}^1 \times \cdots \times \mathcal{X}^d \to \mathbb{F}_2$ *be a function. We say that* cr *is a (d-dimensional)* combinatorial rectangle function *if the truth table of* cr *forms a (d-dimensional) combinatorial rectangle. In other words, for each* $i \in [d]$*, there exist subsets* $\mathcal{S}^i \subseteq \mathcal{X}^i$ *such that* $\mathsf{cr}(x_1, \ldots, x_d) = 1$ *if and only if* $x_i \in \mathcal{S}^i$ *for all* $i \in [d]$*. A combinatorial rectangle function* cr *can be described by* $\hat{\mathsf{cr}} = (\mathcal{S}^1, \ldots, \mathcal{S}^d)$ *of length* $|\hat{\mathsf{cr}}| = O(n)$*, and an evaluation algorithm* E_{CR} *such that* $E_{\mathrm{CR}}(\hat{\mathsf{cr}}, x) = \mathsf{cr}(x)$*.*

Definition 6 (Sum of Combinatorial Rectangles). *Let* $\ell, d \in \mathbb{N}$*. The family of* ℓ*-sum d-dimensional combinatorial rectangle functions is a tuple* $\mathrm{SUMCR}_n^{\ell,d} = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E_n)$ *where* $\mathcal{X}_n = \mathcal{X}_n^1 \times \cdots \times \mathcal{X}_n^d$ *for some sets* $\mathcal{X}_n^1, \ldots, \mathcal{X}_n^d$*,* $P_n = \{\hat{\mathsf{cr}}\}_{\hat{\mathsf{cr}}=(\hat{\mathsf{cr}}_1,\ldots,\hat{\mathsf{cr}}_\ell)}$ *is the set of all* ℓ*-tuples of descriptions of combinatorial rectangle functions with domain* $\mathcal{X}_n$*, and* $E(\hat{\mathsf{cr}}, x) = \sum_{i=1}^{\ell} E_{\mathrm{CR}}(\hat{\mathsf{cr}}_i, x) = \sum_{i=1}^{\ell} \mathsf{cr}_i(x)$*. That is,* $\mathrm{SUMCR}_n^{\ell,d}$ *defines all functions of the form* $f = \mathsf{cr}_1 + \ldots + \mathsf{cr}_\ell$*.*

We next define natural special cases of combinatorial rectangle functions. The first are interval functions which output 1 when the input falls into specified intervals. The second are DNF formulas.

Definition 7 (Interval Functions). *Let* $\ell, d \in \mathbb{N}$ *with* $d|n$*. The family of* ℓ*-sum d-dimensional interval functions is a tuple* $\mathrm{SUMINT}_n^{\ell,d} = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E_n)$ *where*

- $\mathcal{X}_n = \left(\{0,1\}^{n/d}\right)^d$,
- $\mathcal{Y}_n = \mathbb{F}_2$,
- $P_n = \left\{(a_i^j, b_i^j)_{i\in[\ell],j\in[d]} : a_i^j, b_i^j \in \{0,1\}^{n/d}\right\}$, *and*
- $E\left((a_i^j, b_i^j)_{i\in[\ell],j\in[d]}, x\right) = \sum_{i=1}^{\ell} \chi_{\prod_{j=1}^{d}[a_i^j, b_i^j]}(x)$.

In a similar fashion we define $\mathrm{INT}_n^{\ell,d} = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E'_n)$ *to be the family of* ℓ*-union* d*-dimensional interval functions, where*

$$E'_n\left((a_i^j, b_i^j)_{i\in[\ell],j\in[d]}, x\right) = \bigvee_{i=1}^{\ell} \chi_{\prod_{j=1}^{d}[a_i^j, b_i^j]}(x).$$

Moreover, let $\mathrm{INT}_n^{\ell,d} = (\mathcal{X}_n, \mathcal{Y}_n, P'_n, E'_n)$ *be the family of* disjoint ℓ*-union* d*-dimensional interval functions, where* $P'_n \subseteq P_n$ *is restricted to only include cases such that at most a single indicator* $\chi_{\prod_{j=1}^{d}[a_i^j, b_i^j]}$ *outputs* 1 *for a given* x*.*

The function family $\mathrm{SEG}_n^{\ell} \coloneqq \mathrm{DINT}_n^{\ell,1}$ corresponds to a disjoint union of one-dimensional intervals.

Next, we say that $\mathcal{F}_n^{\ell}$ is a *subfamily* of $\mathcal{G}_n^{\ell}$ if their domain and range sets, $\mathcal{X}_n$ and $\mathcal{Y}_n$, match, and any function $f \in \mathcal{F}_n^{\ell}$ can be expressed as a sum (over $\mathcal{Y}_n$) of $O(1)$ functions from $\mathcal{G}_n^{\ell}$.

Proposition 1 (Intervals are Rectangles). $\mathrm{SUMINT}_n^{\ell,d}$ *is a subfamily of* $\mathrm{SUMCR}_n^{\ell,d}$*. In particular, any single interval function with description* $\{(a_i, b_i)\}_{i\in[d]}$ *corresponds to the combinatorial rectangle with description* $\{S_i = \{a_i, a_i + 1, \ldots, b_i\}\}_{i\in[d]}$*.*

Definition 8 (DNF Formulas). *Let* $\ell \in \mathbb{N}$*. The family of functions computed by* ℓ*-sum disjunctive terms is a tuple* $\mathrm{SUMDNF}_n^{\ell} = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E_n)$ *where* $P_n = \{(c_1, \ldots, c_\ell)\}_{c_1,\ldots,c_\ell}$ *consists of all* ℓ*-tuples of disjunctive terms over* n *Boolean variables, and* E_n *is such that* $E_n((c_1, \ldots, c_\ell), (x_1, \ldots, x_n)) = \sum_{i=1}^{\ell} c_i(x_1, \ldots, x_n)$*.* $c_1, \ldots, c_\ell$ *are called the* terms *of the DNF formula.*

In a similar fashion, the family of functions computed by ℓ*-term DNFs is a tuple* $\mathrm{DNF}_n^{\ell} = (\mathcal{X}_n, \mathcal{Y}_n, P_n, E'_n)$ *where* E_n *is such that* $E_n((c_1, \ldots, c_\ell), (x_1, \ldots, x_n)) = \bigvee_{i=1}^{\ell} c_i(x_1, \ldots, x_n)$*.*

Finally, the family of functions computed by ℓ*-term disjoint DNFs is a tuple* $\mathrm{DDNF}_n^{\ell} = (\mathcal{X}_n, \mathcal{Y}_n, P'_n, E'_n)$ *where* $P'_n \subseteq P_n$ *is restricted to only include cases such that at most a single term* c_i *outputs* 1 *for any given* x*.*

Functions computed by decision trees of ℓ leaves can also be computed by ℓ-term disjoint DNF formulas. Therefore the shortcuts we obtain for (disjoint) DNFs apply to decision trees as well.

While the dimension d is not part of the description of DNF formulas over n boolean variables $x_1, \ldots, x_n$, by introducing a interdmediate "dimension" parameter d and partitioning the n variables into d parts, we can represent the DNF formula as a d-dimensional truth table. More concretely, every dimension corresponds to the evaluations of $\frac{n}{d}$ variables. Through this way, we can embed the function into combinatorial rectangles.

Proposition 2 (DNFs are Rectangles). *For any dimension $d \in [n]$, the family* SUMDNF_n^ℓ *is a subfamily of* $\mathrm{SUMCR}_n^{\ell,d}$.

Remark 1 (Disjoint union and general union). The ability to evaluate the sum variants of DNF and INT implies the ability to evaluate the disjoint union because disjoint union can be carried out as a summation. However, the general operation of union is more tricky if the addition is over $\mathbb{F}_2$. It is possible to perform union by (1) having summations over $\mathbb{Z}_m$ for a large enough m such as $m > \ell$, which blows up the input and output share size by a factor of $O(\log \ell)$; or by (2) sacrificing perfect correctness for ϵ-correctness, using random linear combinations, thus multiplying the output share size by $O(\log(1/\epsilon))$. Note that this only works for disjunctions and not for more complex predicates. For instance, for depth-3 circuits we don't have a similar technique.

3.2 Secret sharing

A secret sharing scheme $\mathcal{L} = (\mathsf{Share}, \mathsf{Dec})$ is a tuple of algorithms. Share allows a secret message $s \in K$ to be shared into n parts, $s^1, \ldots, s^n \in K'$ such that they can be distributed among servers $S_1, \ldots, S_n$ in a secure way. Typically, any single share s^j reveals no information about s in the information-theoretic sense. Dec allows authorized server sets to recover s from their respective shares $\{s^j\}$.

We only consider *linear* secret sharing schemes $\mathcal{L} : K \to K'$ in which K and K' are additive groups and the shares satisfy that $\{s^j_{\mathcal{L}} + s'^j_{\mathcal{L}}\}$ is a valid sharing of $s + s'$ under $\mathcal{L}$. We will use linear secret sharing schemes over finite fields and over rings of the form $\mathbb{Z}_m$. Another feature of these schemes that we will require, is that the client's reconstruction algorithm for s is a linear function of (some of) the shares $s^1, \ldots, s^n$. Linear secret sharing schemes can be viewed as *homomorphic secret sharing* schemes, endowed with a linear homomorphism Eval, which we will define more formally in Definition 9.

An *additive* secret-sharing scheme $\mathcal{L}_{\mathrm{add}}$ over an Abelian group splits a secret into random group elements that add up to the secret. For other types of linear secret-sharing, our results will mostly treat them abstractly and will not be sensitive to the details of their implementation; see [12] for formal definitions of the flavors of "Shamir's scheme" and "CNF scheme" we will refer to.

3.3 HSS and PIR

Definition 9 (Information-Theoretic HSS). *An information-theoretic k-*server homomorphic secret sharing scheme *for a function family $\mathcal{F}_n$, or k-HSS for short, is a tuple of algorithms* $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{Share}(x)$*: On input $x \in \mathcal{X}_n$, the sharing algorithm* Share *outputs k input shares, $(x^1, \ldots, x^k)$, where $x^i \in \{0,1\}^{\alpha(N)}$, and some decoding information η.*
- $\mathsf{Eval}(\rho, j, \hat{f}, x^j)$*: On input $\rho \in \{0,1\}^{\gamma(n)}$, $j \in [k]$, $\hat{f} \in P_n$, and the share x^j, the evaluation algorithm* Eval *outputs $y^j \in \{0,1\}^{\beta(N)}$, corresponding to server*

j's share of $f(x)$. *Here* ρ *are public random coins common to the servers and* j *is the label of the server.*

– $\mathsf{Dec}(\eta, y^1, \ldots, y^k)$*: On input the decoding information* η *and* $(y^1, \ldots, y^k)$*, the decoding algorithm* Dec *computes a final output* $y \in \mathcal{Y}_n$*.*

We require the tuple $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ to satisfy correctness and security.

Correctness Let $0 \leq \epsilon < 1$. We say that the HSS scheme is ϵ-correct if for any $f \in \mathcal{F}_n$ and $x \in \mathcal{X}_n$

$$\mathsf{Pr}\left[\mathsf{Dec}\left(\eta, y^1, \ldots, y^k\right) = f(x) : \begin{array}{c} \rho \in_R \{0,1\}^{\gamma(n)} \\ \left(x^1, \ldots, x^k, \eta\right) \leftarrow \mathsf{Share}(x) \\ \forall j \in [k]\ y^j \leftarrow \mathsf{Eval}(\rho, j, \hat{f}, x^j) \end{array}\right] \geq 1 - \epsilon.$$

If the HSS scheme is 0-correct, then we say the scheme is perfectly correct.

Security Let $x, x' \in \mathcal{X}_n$ be such that $x \neq x'$. We require that for any $j \in [k]$ the following distributions are identical

$$\{x^j : (x^1, \ldots, x^k, \eta) \leftarrow \mathsf{Share}(x)\} \equiv \{x'^j : (x'^1, \ldots, x'^k, \eta') \leftarrow \mathsf{Share}(x')\}.$$

For perfectly correct HSS we may assume without loss of generality that Eval uses no randomness and so $\gamma(n) = 0$. In general, we will omit the randomness parameter ρ from Eval for perfectly correct HSS and PIR. Similarly, whenever Dec does not depend on η we omit this parameter from Share and Dec as well.

An HSS is said to be *additive* [21] if Dec simply computes the sum of the output shares over some additive group. This property is useful for composing HSS for simple functions into one for more complex functions. We will also be interested in the following weaker notion which we term *quasiadditive HSS*.

Definition 10 (Quasiadditive HSS). *Let* $\mathit{HSS} = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ *be an HSS for a function family* $\mathcal{F}$ *such that* $\mathcal{Y}_n = \mathbb{F}_2$*. We say that* HSS *is* quasiadditive *if there exists an Abelian group* $\mathbb{G}$ *such that* Eval *outputs elements of* $\mathbb{G}$*, and* $\mathsf{Dec}(y^1, \ldots, y^k)$ *computes an addition* $\tilde{y} = y^1 + \ldots + y^k \in \mathbb{G}$ *and outputs* 1 *if and only if* $\tilde{y} \neq 0$*.*

Definition 11 (PIR). *If the tuple* $\mathit{HSS} = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ *is a perfectly correct* k*-HSS for the function family* ALL_n*, we say that* HSS *is a* k-server private information retrieval scheme*, or* k*-PIR for short.*

Finally, the local computation Eval is modelled by a RAM program.

Definition 12 (Computational shortcut in PIR). *Let* $\mathit{PIR} = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ *be a PIR with share length* $\alpha(N)$*, and* $\mathcal{F}$ *be a function family. We say that* PIR *admits a* strong shortcut *for* $\mathcal{F}$ *if there is an algorithm for* Eval *which runs in quasilinear time* $\tau(N, \ell) = \tilde{O}(\alpha(N) + \beta(N) + \ell)$ *for every function* $f \in \mathcal{F}$*, where* $\ell = |\hat{f}|$ *is the description length of* f*. In similar fashion, we say that* PIR *admits a* (weak) shortcut *for* $\mathcal{F}$ *if there is an algorithm for* Eval *which runs in time* $\tau(N, \ell) = O(\ell \cdot N^{\delta})$*, for some constant* $0 < \delta < 1$*.*

$\mathsf{Share}_{\mathrm{RM}}(x)$:

1. Let $d = k-1$. Divide $x \in \{0,1\}^n$ into d pieces $x = (x_1, \ldots, x_d) \in \left(\{0,1\}^{n/d}\right)^d$.
2. For every $i \in [d]$ compute a unit vector $e_i \in \mathbb{F}_2^{N^{1/d}}$ as $e_i[z] = \begin{cases} 1, & z = x_i \\ 0, & z \neq x_i \end{cases}$.
3. Let $\mathbb{F} = \mathbb{F}_{2^\kappa}$ be a field with $2^\kappa > k$ elements. Let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$ be distinct nonzero field elements. Draw random vectors $r_1, \ldots, r_d \in_R \mathbb{F}^{N^{1/d}}$ and compute $q_i^j := e_i + r_i \alpha_j$ for $i \in [d]$ and $j \in [k]$.
4. The share of each server $j \in [k]$ is $x^j := (q_1^j, \ldots, q_d^j)$. Output $(x^1, \ldots, x^k)$.

$\mathsf{Eval}_{\mathrm{RM}}(j, \hat{f}, x^j = (q_1^j, \ldots, q_d^j))$:

1. Let $\lambda_j := \prod_{\ell \neq j} \alpha_\ell / (\alpha_\ell - \alpha_j)$ be the j'th Lagrange coefficient. Compute

$$\tilde{y}^j = \lambda_j \sum_{(x'_1, \ldots, x'_d) \in \{0,1\}^n} f(x'_1, \ldots, x'_d) \prod_{i=1}^{d} (q_i^j)[x'_i]$$

2. Output $y^j = \sigma(\tilde{y}^j)$, where $\sigma : \mathbb{F} \to \mathbb{F}_2$ is a homomorphism with respect to addition such that $\sigma(z) = z$ for $z \in \mathbb{F}_2$.

$\mathsf{Dec}_{\mathrm{RM}}(y^1, \ldots, y^k)$: Output $y = y^1 + \ldots + y^k$.

HSS Parameters: Input share size $\alpha(N) = O(N^{1/d})$, output share size $\beta(N) = 1$.

Fig. 1. The scheme $\mathsf{PIR}^k_{\mathsf{RM}}$.

4 Shortcuts for Reed-Muller PIR

Let $3 \leq k \in \mathbb{N}$ and $d = k - 1$ be constants. The k-server Reed-Muller based PIR scheme $\mathsf{PIR}^k_{\mathsf{RM}} = (\mathsf{Share}_{\mathrm{RM}}, \mathsf{Eval}_{\mathrm{RM}}, \mathsf{Dec}_{\mathrm{RM}})$ is presented in Figure 1.

We observe that, in k-server Reed-Muller PIR $\mathsf{PIR}^k_{\mathsf{RM}}$, the sum of products

$$\sum_{(x'_1, \ldots, x'_d) \in \{0,1\}^n} f(x'_1, \ldots, x'_d) \prod_{i=1}^{d} (q_i^j)[x'_i]$$

can be written as a product of sums if f is a combinatorial rectangle function. Consequently $\mathsf{PIR}^k_{\mathsf{RM}}$ admits a computational shortcut for d-dimensional combinatorial rectangles, which gives rise to shortcuts for intervals and DNFs as they are special cases of combinatorial rectangles (Propositions 1 and 2).

Lemma 2. $\mathsf{PIR}^k_{\mathsf{RM}}$ *admits a strong shortcut for the function family of single d-dimensional combinatorial rectangle, i.e.,* $\mathrm{SUMCR}_n^{1,d}$. *More concretely,* $\tau(N, \ell) = O(\alpha(N)) = O(N^{1/d})$.

Proof. Naturally, the client and server associate $x = (x_1, \ldots, x_d)$ as the input to the funcions f from $\mathrm{SUMCR}_n^{1,d}$. Let $\hat{f} = \hat{\mathsf{cr}} = \{\mathcal{S}_1, \ldots, \mathcal{S}_d\}$ be the combinatorial

rectangle representing f. Given $\hat{f}$, the computation carried out by server j is

$$\mathsf{Eval}_{\mathrm{RM}}(j, \hat{f}, x^j = (q_1^j, \ldots, q_d^j)) = \sigma\left(\lambda_j \sum_{(x'_1,\ldots,x'_d)\in \mathcal{S}_1\times\ldots\times\mathcal{S}_d} \prod_{i=1}^{d} q_i^j[x'_i]\right) \tag{3}$$

$$= \sigma\left(\lambda_j \prod_{i=1}^{d} \sum_{x'_i\in\mathcal{S}_i} q_i^j[x'_i]\right) \tag{4}$$

If the server evaluates the expression using Equation (3) the time is $O(N)$, but if it instead uses Equation (4) the time is $O(d\max_i\{|S_i|\}) = O(2^{\frac{n}{d}}) = O(\alpha(N))$.

Theorem 9. $\mathsf{PIR}^k_{\mathsf{RM}}$ *admits a weak shortcut for the function family* $\mathrm{SUMCR}^{\ell,d}_N$. *More concretely,* $\tau(N,\ell) = O(\ell\alpha(N)) = O(\ell N^{1/d})$. *The same shortcut exists for decision trees with* ℓ *leaves, or, more generally,* SUMDNF^{ℓ}_n *and* DDNF^{ℓ}_n.

Proof. This is implied by Lemma 2, by noting that $f = \mathsf{cr}_1 + \ldots \mathsf{cr}_\ell$ over the common input x. In particular, the final Eval algorithm makes ℓ calls to the additive HSS given by Lemma 2, so the running time is $O(\ell\alpha(N)) = O(\ell 2^{\frac{n}{d}})$.

An algorithm, presented in the full version, improves the efficiency of Theorem 9 for decision trees to $\tilde{O}(\alpha(n) + \ell\cdot\alpha(n)^{1/3})$.

4.1 Intervals and Convex Shapes

By Proposition 1, one obtains weak shortcuts for d-dimensional intervals. In fact, one can obtain *strong shortcuts* by the standard technique of precomputing the prefix sums in the summation Equation (4).

Theorem 10. $\mathsf{PIR}^k_{\mathsf{RM}}$ *admits a strong shortcut for the function family* $\mathrm{SUMINT}^{\ell,d}_n$. *More concretely,* $\tau(N,\ell) = O(\alpha(N)+\ell) = O(N^{1/d}+\ell)$. *The same shortcut applies to* $\mathrm{DINT}^{\ell,d}_n$.

Segments and Low-Dimensional Intervals Every segment can be split into at most $(2d-1)$ d-dimensional intervals. The splitting (deferred to the full version) works by comparing the input $x\in\{0,1\}^n$ with the endpoints $a, b\in(\{0,1\}^{n/d})^d$ in a block-wise manner.

Theorem 11. $\mathsf{PIR}^k_{\mathsf{RM}}$ *admits a strong shortcut for the function families* SEG^{ℓ}_n. *Generally, for every integer* $d'|d$, $\mathsf{PIR}^k_{\mathsf{RM}}$ *admits a strong shortcut for the function families* $\mathrm{DINT}^{d',\ell}_n$ *(or* $\mathrm{SUMINT}^{d',\ell}_n$*). More concretely,* $\tau(N,\ell) = O(N^{1/d}+\ell)$.

Shortcut for Convex Shapes At a high level, convex body functions are functions whose preimage of 1 forms a convex body in the d-dimensional cube $\mathcal{X}_n := (\{0,1\}^{n/d})^d$. The following theorem follows from the fact that we can efficiently split a d-dimensional convex body into $O(N^{(d-1)/d})$ d-dimensional intervals in a "Riemann-sum" style.

Theorem 12 (Convex bodies, Informal). *There is a perfectly correct k-server HSS for the function class of ℓ-unions of convex shapes with $\alpha(n) = O(N^{1/(k-1)})$, $\beta(n) = 1$ and $\tau(n) = O(\ell N^{(k-2)/(k-1)})$.*

We show that the bound is essentially the best achievable by splitting the shape into union of intervals. Finally, we show that on the other hand, for more regular shapes like circles, strong shortcuts are possible if one settles for an approximated answer. Detailed discussion of such results are deferred to the full version.

Theorem 13 (Circle approximation, Informal). *There is a perfectly correct k-server HSS for the function class of ℓ-unions of ϵ-approximations of circles with $\alpha(n) = O(N^{1/(k-1)})$, $\beta(n) = 1$ and $\tau(n) = O(\alpha(n) + \frac{1}{\epsilon}\ell)$.*

4.2 Compressing Input Shares

The scheme $\mathsf{PIR}^3_{\mathsf{RM}}$ described above can be strictly improved by using a more dense encoding of the input. This results in a modified scheme $\mathsf{PIR}^3_{\mathsf{RM}'}$ with $\alpha'(N) = \sqrt{2} \cdot N^{1/2}$, a factor $\sqrt{2}$ improvement over $\mathsf{PIR}^3_{\mathsf{RM}}$. This is the best known 3-server PIR scheme with $\beta = 1$ (up to lower-order additive terms [11]). In the full version, we show that with some extra effort, similar shortcuts apply also to the optimized $\mathsf{PIR}^3_{\mathsf{RM}'}$.

4.3 Negative Results for RM PIR

Although we have shortcuts for disjoint DNF formulas, similar shortcut for more expressive families with *counting hardness* is unlikely. The idea is similar in spirit to [52, Claim 5.4]. The lower bounds for $\mathsf{PIR}^3_{\mathsf{RM}}$ also hold for $\mathsf{PIR}^3_{\mathsf{RM}'}$.

Theorem 14. *Let $\mathcal{F}$ be a function family for which $\mathsf{PIR}^k_{\mathsf{RM}}$ admits a weak shortcut with $\tau(N, \ell) = T$. Then, there exists an algorithm $\mathsf{Count}_2 : P_n \to \mathbb{F}_2$ running in time $O(T + \left|\hat{f}\right|)$, that when given $\hat{f} \in P_n$, computes the parity of $|\{x \in \mathcal{X}_n : f(x) = 1\}|$.*

Proof. Recall that the server computes the following expression in $\mathsf{PIR}^k_{\mathsf{RM}}$:

$$\sigma\left(\lambda_j \sum_{(x'_1,\ldots,x'_{k-1})\in\mathcal{X}^n} f(x'_1,\ldots,x'_{k-1}) \prod_{i=1}^{k-1} (q_i^j)[x'_i]\right).$$

To compute the required parity, instead of using $e_1, \ldots, e_n$ in the original $\mathsf{Share}_{\mathsf{RM}}$ in step 3 (see Figure 1), we use the vectors $1^{N^{1/d}}, \ldots, 1^{N^{1/d}}$, *i.e.*, the all-one vectors. After calling Eval on all the respective shares and decoding the output, one obtains

$$\sum_{(x'_1,\ldots,x'_{k-1})\in\{0,1\}^n} f(x'_1,\ldots,x'_{k-1}) = |\{x \in \mathcal{X}_n : f(x) = 1\}| \pmod 2.$$

The total time of the algorithm is $O(T + \left|\hat{f}\right|)$.

We recall the following conjecture commonly used in complexity theory.

Conjecture 2 (Strong Exponential Time Hypothesis (SETH)). SAT cannot be decided with high probability in time $O(2^{(1-\epsilon)n})$ for any $\epsilon > 0$.

By the isolation lemma from [25], SETH is known to imply that $\oplus\mathsf{SAT}$, which is similar to SAT except that one need to compute the parity of the number of satisfying assignments, cannot by solved in time $O(2^{(1-\epsilon)n})$. The number of satifying assignemnts to a CNF formula equals $2^n - r$, where r is the number of satisfying assignments to its negation. Since the negation of a CNF formula is in DNF, $\oplus$DNF cannot be decided in $O(2^{(1-\epsilon)n})$ as well. Therefore we have the following corollary.

Corollary 2. *For any k, there exists a polynomially bounded ℓ such that $\mathsf{PIR}^k_{\mathsf{RM}}$ does not admit a weak shortcut for the function family DNF^ℓ_n, unless SETH fails.*

Proof. By Theorem 14, if there is a weak shortcut for any polynomially bounded ℓ, *i.e.*, an algorithm computing Eval for any function in DNF^ℓ_n in time $O(N^{1-\epsilon})$, then one can decide $\oplus$DNF in time $O(N^{1-\epsilon})$.

Note that the hardness for DNF^ℓ_n is not contradictory to the fact that larger field size or random linear combinations help evaluating general DNFs (see Remark 1) because our proof heavily relies on the fact that we work over a small field (which has several efficiency benefits) and that the shortcut is deterministic.

Conjecture 3 (Exponential Time Hypothesis (ETH)). SAT requires time $O(2^{\delta n})$, for some $\delta > 0$, to be decided with high probability.

In a similar fashion, assuming the ETH, we can obtain the weaker result that strong shortcuts are impossible given k is large, namely when $k > \frac{1}{\delta}$.

Corollary 3. *Assume ETH. For some large enough k and some polynomially bounded ℓ, $\mathsf{PIR}^k_{\mathsf{RM}}$ does not admit a strong shortcut for the function family DNF^ℓ_n.*

5 On Shortcuts for Matching Vector PIR

Matching vectors (MV) based PIR schemes in the literature can be cast into a template due to [12]. As described in the introduction, this template has two ingredients: (1) a *matching vector family*; (2) a *share conversion*. A complete specification is given in the full version.

We describe the server computation in more detail, in particular, we present the structure of the matching vector family on which MV PIR is based. In $\mathsf{PIR}^k_{\mathsf{MV},\mathsf{SC}}$ each server j is given as input $x^j \in \mathbb{Z}^h_m$ which is a secret share of u_x. Then, for every $x' \in [N]$, the server j homomorphically obtains $y_{j,x,x'}$ which is the j-th share of $\langle u_x, v_{x'} \rangle$. Next, each server j computes a response

$$\sum_{x' \in [N]} f(x')\mathsf{SC}(j, y_{j,x,x'}).$$

Therefore, for the all-1 database ($f(x) = 1$), for every S-matching vector family MV and share conversion scheme SC from $\mathcal{L}$ to $\mathcal{L}'$ we can define the $(\mathsf{MV}, \mathsf{SC})$-counting problem, $\#(\mathsf{MV}, \mathsf{SC})$, see Definition 1.

We consider $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC})$, where $\mathsf{MV}^w_{\mathrm{Grol}}$ is a matching vectors family due to Grolmusz [40], which is used in all third-generation PIR schemes, which we present in Fact 1, and $\mathsf{SC} \in \{\mathsf{SC}_{\mathrm{Efr}}, \mathsf{SC}_{\mathrm{BIKO}}\}$.

$\#(\mathsf{MV}, \mathsf{SC})$ displays a summation of converted shares of inner products. The actual computation carried out is determined by the structure of $v_{x'}$ and hence the instance of the MV used. Here we describe the *graph-based* matching vector family, first given in [40].

Instantiation of Grolmusz's family There is an explicitly constructable S-Matching Vector family for $m = p_1 p_2$ with $\alpha(N) = N^{o(1)}$ based on the intersecting set family in [50] for the *canonical set* $S = S_m = \{(0,1),(1,0),(1,1)\} \subseteq \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2}$ (in Chinese remainder notation). Here we give a more detailed description of their structure in the language of hypergraphs.

Fact 1 (The parameterized $\mathsf{MV}^w_{\mathrm{Grol}}$, modified from [40]) *Let $m = p_1 p_2$ where $p_1 < p_2$ are distinct primes. For any integer r and parameter function $w(r)$, one can construct an S-matching vector family $\{u_x, v_x \in \mathbb{Z}_m^h\}_{x \in [N]}$ where $N = \binom{r}{w(r)}$ and $h = \binom{r}{\leq d}$ for $d \leq p_2\sqrt{w(r)}$. Moreover, the construction is* hypergraph-based *in the following sense:*

Let $[r]$ be the set of vertices. Every index $x \in [N]$ corresponds to a set $T_x \subset [r]$ of $w(r)$ nodes. The vector v_x has entires in $\{0,1\}$ and its coordinates are labelled with $\zeta \subset [r]$ which are hyperedges of size at most d nodes. Moreover, $v_x[\zeta] = 1$ iff the vertices of the hyperedge ζ are all inside T_x. Therefore the inner product can be evaluated as

$$\langle u_x, v_{x'} \rangle = \sum_{\zeta \subseteq T_{x'}, |\zeta| \leq d} u_x[\zeta] = \sum_{\zeta \subseteq T_x, |\zeta| \leq d} u_{x'}[\zeta] = \sum_{\zeta \subseteq T_x \cap T_{x'}, |\zeta| \leq d} 1.$$

In other words, the inner product is carried out by a summation over all the hyperedges lying within a given vertex subset $T_{x'}$. Under this view, we will call $|T_{x'}| = w(r)$ the clique size parameter.

By setting $\mathsf{MV}^w_{\mathrm{Grol}}$ with $w = \Theta(\sqrt{r})$, we obtain from Fact 1 and the definition of $\mathsf{PIR}^k_{\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}}$, a PIR scheme with $\alpha(N) = 2^{O(2p_2\sqrt{n \log n})}$, which is state of the art in terms of asymptotic communication complexity. We prove Fact 1 in the full version.

5.1 A Reduction From a Subgraph Counting Problem for $\mathsf{SC}_{\mathrm{Efr}}$

In this section we relate the server computation to a subgraph counting problem. For this we rely on the hypergraph-based structure of the matching vector family, in combination with the share conversion $\mathsf{SC}_{\mathrm{Efr}}$. More concretely, we relate $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}_{\mathrm{Efr}})$ to the problem $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$, see Definition 2 and the preceding discussion.

We prove the following which relates obtaining computational shortcuts for $\mathsf{PIR}^k_{\mathsf{MV},\mathsf{SC}}$ to counting induced subgraphs.

Lemma 3 (Hardness of $(\mathbf{MV}^w_{\mathrm{Grol}}, \mathbf{SC}_{\mathrm{Efr}})$-counting). *If $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}_{\mathrm{Efr}})$ can be computed in $N^{o(1)}$ $\left(= r^{o(w)}\right)$ time, then $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ can be decided in $r^{o(w)}$ time, for any nondecreasing function $w\colon \mathbb{N} \to \mathbb{N}$.*

In particular, if we can show $\oplus\textsc{IndSub}(\Phi_{511,0}, \Theta(\sqrt{r}))$ cannot be decided in $r^{o(\sqrt{r})}$ time under some complexity assumption, it will imply that $\mathsf{PIR}^k_{\mathsf{MV}^{\Theta(\sqrt{r})}_{\mathrm{Grol}},\mathsf{SC}_{\mathrm{Efr}}}$ does not admit strong shortcuts for the all-1 database under the same assumption, as $\alpha(N) = N^{o(1)}$ holds and $\tau(N,\ell) = N^{o(1)}$ is impossible.

Proof (Proof of Lemma 3). Let $m = 511$. Recall that $N = \binom{r}{w}$ and $h = \binom{r}{\le d}$ where $d \le p_2\sqrt{w}$. Suppose A is an algorithm solving $\#(\mathsf{MV}^w_{\mathrm{Grol}}, \mathsf{SC}_{\mathrm{Efr}})$ with these parameters that runs in time $N^{o(1)} = r^{o(w)}$. By definition of $\mathsf{Share}_{\mathrm{Efr}}$, the input to A is a vector $x^j \in \mathbb{Z}_m^h$. To homomorphically obtain a share of $\langle u_x, v_{x'}\rangle$, where x is the client's input, the server first computes $\langle x^j, v_{x'}\rangle$. For any instance G in $\oplus\textsc{IndSub}(\Phi_{m,0}, w)$ with $|V(G)| = r$, we define the following vector $q \in \mathbb{Z}_m^h$: for every hyperedge ζ where $|\zeta| \le d$,

$$q[\zeta] = \begin{cases} 0 & \text{if } \zeta \notin E(G) \\ 1 & \text{if } \zeta \in E(G). \end{cases} \tag{5}$$

Note that for any $|\zeta| \neq 2$ we have $q[\zeta] = 0$. By Fact 1 and how q is constructed, for every $x' \in [N]$,

$$\langle q, v_{x'}\rangle = \sum_{\zeta \subset T_{x'}, |\zeta| \le d} q[\zeta] = \sum_{\zeta \subset T_{x'}, \zeta \in E(G)} 1.$$

Therefore the value of the inner product is the number of edges in the subgraph induced by the nodes in $T_{x'}$. For $\ell = 1, \ldots, (m-1)$, we feed $\ell \cdot q$ into the algorithm A. The output will be

$$\begin{aligned} \sum_{x' \in [N]} \mathsf{SC}_{\mathrm{Efr}}(j, \langle \ell \cdot q, v_{x'}\rangle) &= \sum_{x' \in [N]} a_j \gamma^{\langle \ell \cdot q, v_{x'}\rangle} = a_j \sum_{x' \in [N]} \gamma^{\ell \langle q, v_{x'}\rangle} \\ &= a_j \sum_{b \in \{0,\ldots,m-1\}} \sum_{x' : \langle q, v_{x'}\rangle = b} \gamma^{b\ell} \\ &= a_j \sum_{b \in \{0,\ldots,m-1\}} c_b (\gamma^\ell)^b, \end{aligned}$$

where $c_b \in \{0,1\}$ (recall that the field $\mathbb{F}_{2^9}$ has characteristic 2) is the parity of the number of induced w-subgraphs, whose number of edges is congruent to b modulo m. This is because c_b counts the number of elements in the set $\{x' \in [N] : \langle q, x'\rangle = b\} = \{x' \in [N] : \sum_{\zeta \subset T_{x'}, \zeta \in E(G)} 1 = b\}$. Consequently, the bit c_0 is the answer bit to the problem $\oplus\textsc{IndSub}(\Phi_{m,0}, w)$. Note that after each call to A, we obtain evaluation of the degree-$(m-1)$ polynomial $Q(\Gamma) = a_j \sum_{b \in \{0,\ldots,m-1\}} c_b \Gamma^b$

at $\Gamma = \gamma^\ell$. Since the points $\{\gamma^\ell\}_{\ell=0}^{m-1}$ are distinct, we can perform interpolation to recover c_b for any $b \in \{0, \ldots, m-1\}$. In particular, we can compute the desired bit c_0. The overall running time is $O(m^2) + mr^{o(w)} = r^{o(w)}$.

In the full version, we show that a similar reduction holds for $\mathsf{SC}_{\mathrm{BIKO}}$ as well, except that we consider the problem $\oplus\textsc{IndSub}(\Phi_{6,4}, w)$.

5.2 Hardness of Subgraph Counting

As described in Section 2.2, we have the following plausible conjecture, and it turns out that its hardness can be based on ETH for a suitable choice of parameter.

Conjecture 4 (Hardness of counting induced subgraphs). $\oplus\textsc{IndSub}(\Phi_{m,\Delta}, w)$ cannot be decided in $r^{o(w)}$ time, for any integers $m \geq 2$, $0 \leq \Delta < m$, and for every function $w(r) = O(r^c)$, $0 \leq c < 1$.

Note that Conjecture 4 does not rule out *weak* shortcuts. However, it seems that even weak shortcuts would be difficult to find when instantiated with matching vectors from Fact 1. Indeed, for the related problem of hyperclique counting, algorithms which are faster than the naïve one are known only for the special case when hyperedges are *edges* (e.g.[4]).

Basing on ETH. Proving Conjecture 4 is difficult as it is a fine-grained lower bound. However, by assuming ETH, we can prove Conjecture 4 partially, in the sense that for a specific choice of $w(r)$, the lower bound does hold.

Lemma 4. *There is an efficiently computable function $w(r) = \Theta(\log r / \log\log r)$, such that if $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ or $\oplus\textsc{IndSub}(\Phi_{6,4}, w)$ can be decided in $r^{o(w(r))}$ time, then ETH fails.*

Proof. This follows from $\mathsf{ETH} \overset{\text{Lemma 5}}{\leq} \textsc{Clique}(k(r)) \overset{\text{Lemma 6}}{\leq} \oplus\textsc{IndSub}(\Phi, w)$ for $\Phi \in \{\Phi_{511,0}, \Phi_{6,4}\}$.

Next, we sketch how to perform the steps of the reduction in the proof of Lemma 4.

Reducing clique decision to ETH. Let $\textsc{Clique}(k(r))$ be the problem that, given a graph G with r nodes, decide whether a clique of size $k(r)$ exists in G. As a direct corollary of [28, Theorem 5.7], we have the following lemma.

Lemma 5. *There is an efficiently computable function $k(r) = \Theta(\log r / \log\log r)$, such that if $\textsc{Clique}(k(r))$ can be solved in $r^{o(k(r))}$ time, then ETH fails.*

Reducing induced subgraph counting to clique decision By reproducing the reduction in [42], we have the following (proofs deferred to the full version).

Lemma 6. *Let $k(r) = \Theta(\log r / \log\log r)$ as in Lemma 5. Then, there is an efficiently computable size parameter $w(r) = \Theta(\log r / \log\log r)$ such that if $\oplus\textsc{IndSub}(\Phi_{511,0}, w)$ or $\oplus\textsc{IndSub}(\Phi_{6,4}, w)$ can be decided in $r^{o(w(r))}$ time, then one can decide $\textsc{Clique}(k(r))$ in $r^{o(k(r))}$ time.*

While Lemma 5 could be proven to hold for $k(r) = \Theta(\sqrt{r})$ as well, as discussed in Section 2.2, by reproducing the reduction in [42], Lemma 6 only holds for $k(r) = o(\log r)$.

Hardness of subgraph counting Finally, our main theorem follows from Conjecture 4 and Lemmas 3 and 4. To this end, denote by MV^* the family $\mathsf{MV}^w_{\mathrm{Grol}}$ obtained by instantiating $w(r)$ with this specific parameter.

Theorem 15. *$\#(\mathit{MV}^*, \mathit{SC})$ cannot be computed in $N^{o(1)}$ $\left(= r^{o(w)}\right)$ time, unless* ETH *fails. Moreover, assuming Conjecture 4, the same holds for $\mathsf{MV}^w_{\mathrm{Grol}}$ with $w = \Theta(\sqrt{r})$. Here SC is either $\mathsf{SC}_{\mathrm{Efr}}$ or $\mathsf{SC}_{\mathrm{BIKO}}$.*

6 Concrete Efficiency

While this paper deals with *computational* shortcuts, in this section we will make comparisons exclusively with respect to *communication*. The main reason we compare communication is that for our main positive results, computation scales at most quasi-linearly with the size of the inputs, and thus is essentially the best one can hope for. Moreover, it is hard to make exact "apples to apples" comparisons for computation (what are the units?) Perhaps most importantly, for the problems to which our positive results apply (e.g., unions of convex shapes), the (asymptotic and concrete) computational efficiency of our schemes dominate those of competing approaches (FHE, brute-force PIR, garbled circuits, GMW-style protocols). Due to the concrete inefficiency of HSS from generic composition of PIRs, we will focus exclusively on HSS from shortcuts for $\mathsf{PIR}^k_{\mathsf{RM}}$.

Cryptographic Share Compression In the full version we describe a simple method [33] to compress the queries of $\mathsf{PIR}^k_{\mathsf{RM}} = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$, at the cost of making the scheme only computationally secure, utilizing share conversion from Shamir secret sharing to CNF secret sharing (c.f. [12] for relevant definitions).

Communication Complexity In Table 1 we compare the communication complexity for unions of disjoint two dimensional intervals. For two dimensional intervals, FSS requires queries of length $O(\lambda(\log N)^2)$ [19].

It is worth mentioning that private geographical queries were already considered in [58]. However, there the two dimensional plane is tesselated with overlapping shapes of the same size, which reduces the problem to the task of

Domain size	Reed-Muller[29] ($k = 3$)	Reed-Muller[29] ($k = 4$)	Reed-Muller[29] ($k = 5$)	FSS[19] ($k = 2$)
2^{10}	0.05 KB	0.05 KB	0.06 KB	3.1 KB
2^{15}	0.1 KB	0.1 KB	0.1 KB	7.0 KB
2^{20}	0.6 KB	0.2 KB	0.3 KB	12.5 KB
2^{25}	2.9 KB	0.5 KB	0.4 KB	19.5 KB
2^{30}	16.1 KB	1.3 KB	0.6 KB	28.1 KB
2^{35}	90.6 KB	3.7 KB	1.1 KB	38.3 KB
2^{40}	512.1 KB	11.5 KB	2.2 KB	50.0 KB
2^{70}	16.0 GB	11.3 MB	362.3 KB	153.1 KB

Table 1. Total communication complexity for the task where the client holds a secret index x in a grid $[\sqrt{N}] \times [\sqrt{N}]$ and it wishes to privately learn (with security threshold $t = 1$) if it is contained in a collection of ℓ two dimensional intervals held by k servers. The computational cost for FSS and Reed-Muller is $\tilde{O}(\mathsf{comm} + \ell)$, where comm is the communication complexity. The latter is obtained via our shortcuts. Note that for $k = 4$ the aforementioned computational cost is obtainable only when considering grids with dimensions $[N^{1/3}] \times [N^{2/3}]$. For grids with dimensions $[\sqrt{N}] \times [\sqrt{N}]$ the computational cost becomes $\tilde{O}(\mathsf{comm} + \ell\sqrt{\mathsf{comm}})$. See [19, Corollary 3.20] for how the numbers in last column were computed. Share compression was applied to Reed-Muller.

evaluating multipoint functions. Therefore, this approach can be seen as a simply reducing the size of the problem. In contrast, here we allow for a better tradeoff between precision and computation. Our solution is more expressive, as it allows for shapes of high and low precision simultaneously.

Larger Security Threshold In this section we consider the applicability of our PIR-based HSS to security models with larger security threshold. Specifically, we will consider the case where we allow at most *two* colluding servers. However, lending to its PIR backbone, our HSS constructions scale well for higher security thresholds.

Indeed, there is an analogue of $\mathsf{PIR}^k_{\mathsf{RM}}$ with 2 security threshold, such that for $O(\sqrt{N})$ and $O(N^{1/3})$ total communication, the number of required servers is 5 and 7, respectively. Moreover, this PIR scheme retains all the computational shortcuts of $\mathsf{PIR}^k_{\mathsf{RM}}$ and its shares can be compressed as well. Alternatively, employing multiparty FSS [17] (for multipoint functions) requires only 3 servers. However, in stark contrast to two party FSS, multiparty FSS requires $O(\lambda\sqrt{N})$ total communication. Moreover, it is not clear how to obtain an FSS for one dimensional intervals in this setting, let alone two dimensional intervals. We conclude our HSS wins by two orders of magnitude.

Another approach to increase the security threshold of FSS is via the generic tensoring technique of [7], which preserves the communication complexity. Nevertheless, this scales worse with larger security threshold t, requiring 2^t servers, compared to $2t + 1$ servers via Reed-Muller PIR. Furthermore, this approach is not computationally efficient, requiring $O(N)$ computation. We provide a description of the tensoring of [7] in the full version.

In Table 2 we compare the communication complexity of FSS with our HSS for the simple task of PIR, as more expressive function families are unavailabe for higher security thresholds for FSS.

Domain size	Reed-Muller[29] $(t = 2, k = 5)$	FSS[19] $(t = 2, k = 3)$
2^{10}	0.2 KB	3.0 KB
2^{15}	0.6 KB	17.0 KB
2^{20}	1.2 KB	96.0 KB
2^{25}	4.7 KB	543.1 KB
2^{30}	24.4 KB	3.0 MB
2^{35}	136.2 KB	17.0 MB
2^{40}	768.4 KB	96.0 MB
2^{70}	24.0 GB	3.0 TB

Table 2. Total communication complexity for the task where the client holds a secret index x in $[N]$ and it wishes to privately learn (with security threshold $t = 2$) if its contained in a collection of ℓ points in $[N]$ held by k servers. The computational cost for FSS and Reed-Muller is $\tilde{O}(\mathsf{comm} + \ell)$, where comm is the communication complexity. Data for FSS was obtained from [17, Theorem 7]. Share compression was applied to Reed-Muller.

Other settings In the full version, we show how to make our schemes more efficient whenever the payload size is small (a few bits), by basing our shortcuts on a "balanced" variant of $\mathsf{PIR}^k_{\mathsf{RM}}$, proposed by Woodruff and Yekhanin [60]. In addition, we discuss our schemes in the context of distributed share generation and argue that our schemes are more "MPC-friendly" than the FSS-based alternative.

References

1. Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *IEEE Symposium on Security and Privacy*, 2018.
2. Benny Applebaum, Thomas Holenstein, Manoj Mishra, and Ofer Shayevitz. The communication complexity of private simultaneous messages, revisited. In *EUROCRYPT 2018, Part II*, pages 261–286, 2018.
3. Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *CCS*, 2016.
4. Per Austrin, Petteri Kaski, and Kaie Kubjas. Tensor network complexity of multilinear maps. In *ITCS*, 2019.
5. Marshall Ball, Justin Holmgren, Yuval Ishai, Tianren Liu, and Tal Malkin. On the complexity of decomposable randomized encodings, or: How friendly can a garbling-friendly PRF be? In *ITCS 2020*, pages 86:1–86:22, 2020.

6. Omer Barkol and Yuval Ishai. Secure computation of constant-depth circuits with applications to database search problems. In Victor Shoup, editor, *CRYPTO 2005*, pages 395–411.
7. Omer Barkol, Yuval Ishai, and Enav Weinreb. On locally decodable codes, self-correctable codes, and t-private PIR. In *Algorithmica*, 2010.
8. Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 1983.
9. Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In *CRYPTO '90*, pages 62–76, 1990.
10. Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *TCC*, 2014.
11. Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *Journal of Computer and System Sciences*, 2005.
12. Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and I. N. Orlov. Share conversion and private information retrieval. *IEEE 27th Conference on Computational Complexity*, 2012.
13. Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the O(nˆ1/(2k-1)) barrier for information-theoretic private information retrieval. In *FOCS*, 2002.
14. Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In *CRYPTO*, 2000.
15. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.
16. Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of A secret sharing. In Andrew M. Odlyzko, editor, *CRYPTO '86*, pages 251–260, 1986.
17. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT*, 2015.
18. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO*, 2016.
19. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, 2016.
20. Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *TCC 2019, Part I*, pages 341–371, 2019.
21. Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *ITCS*, 2018.
22. Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT*, 2019.
23. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, 2019.
24. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.
25. Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-SAT: An isolation lemma for k-CNFs. *Journal of Computer and System Sciences*, 2008.
26. Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *IWPEC*, 2009.
27. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, 1988.

28. Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 2006.
29. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *J. ACM*, 1998.
30. Joshua N. Cooper, Robert B. Ellis, and Andrew B. Kahng. Asymmetric binary covering codes. *J. Comb. Theory, Ser. A*, 2002.
31. Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *IEEE Symposium on Security and Privacy*, 2015.
32. Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In *EUROCRYPT 2019, Part II*, pages 473–503, 2019.
33. Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *TCC*, 2005.
34. Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, 2000.
35. Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure mpc, with or without preprocessing. In *CRYPTO 2019, Part II*, pages 61–84, 2019.
36. Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *CRYPTO 2017, Part I*, pages 167–187, 2017.
37. Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael A. Raskin. On the communication required for unconditionally secure multiplication. In *CRYPTO 2016, Part II*, pages 459–488, 2016.
38. Deepesh Data, Manoj Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In *CRYPTO 2014, Part II*, pages 199–216, 2014.
39. Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, 2016.
40. Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. In *FOCS*, 2010.
41. Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In *FOCS*, 2015.
42. Julian Dörfler, Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting induced subgraphs: An algebraic approach to #W[1]-hardness. *MFCS*, 2019.
43. Klim Efremenko. 3-query locally decodable codes of subexponential length. In *STOC*, 2009.
44. Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from Paillier encryption. In *Provable Security*, 2017.
45. Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *STOC 1994*, pages 554–563, 1994.
46. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.
47. Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC*, 2019.
48. Niv Gilboa and Yuval Ishai. Compressing cryptographic resources. In *CRYPTO*, 1999.
49. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, 1987.

50. Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Combinatorica*, 2000.
51. Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath T. V. Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and private media consumption with popcorn. In *USENIX*, 2016.
52. Prahladh Harsha, Yuval Ishai, Joe Kilian, Kobbi Nissim, and Srinivasan Venkatesh. Communication versus computation. In *ICALP*, 2004.
53. Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC 2013*, pages 600–620, 2013.
54. Russell W. F. Lai, Giulio Malavolta, and Dominique Schröder. Homomorphic secret sharing for low degree polynomials. In *ASIACRYPT*, 2018.
55. Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private information retrieval for everyone. *PoPETs*, 2016.
56. Marc Roth. *Counting Problems on Quantum Graphs*. PhD thesis, 2019.
57. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 1979.
58. Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *USENIX*, 2017.
59. Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, 2018.
60. David P. Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *CCC*, 2005.
61. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.
62. Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *STOC*, 2007.
63. Sergey Yekhanin. *Private information retrieval*. 2010.