On the Cryptographic Complexity of the Worst Functions

Amos Beimel* Yuval Ishai† Ranjit Kumaresan† Eyal Kushilevitz†

Abstract

We study the complexity of realizing the "worst" functions in several standard models of information-theoretic cryptography. For each of these models, we obtain the first solution whose complexity is sublinear in the relevant domain size. In particular, for the case of security against passive adversaries, we obtain the following main results.

- **OT complexity of secure two-party computation.** Every function $f : [N] \times [N] \to \{0,1\}$ can be securely evaluated using $\widetilde{O}(N^{2/3})$ invocations of an oblivious transfer oracle. A similar result holds for securely sampling a uniform pair of outputs from a set $S \subseteq [N] \times [N]$.
- **Correlated randomness complexity of secure two-party computation.** Every function $f : [N] \times [N] \to \{0,1\}$ can be securely evaluated using $2^{\widetilde{O}(\sqrt{\log N})}$ bits of correlated randomness.
- **Communication complexity of private simultaneous messages.** Every function $f : [N] \times [N] \to \{0,1\}$ can be securely evaluated in the non-interactive model of Feige, Kilian, and Naor (STOC 1994) with messages of length $O(\sqrt{N})$.
- **Share complexity of forbidden graph access structures.** For every graph G on N nodes, there is a secret-sharing scheme for N parties in which each pair of parties can reconstruct the secret if and only if the corresponding nodes in G are connected, and where each party gets a share of size $\widetilde{O}(\sqrt{N})$.

For all of these problems, the worst-case complexity of the best previous solutions was $\Omega(N/\log N)$. The above results are obtained by applying general transformations to variants of private information retrieval (PIR) protocols from the literature, where different flavors of PIR are required for different applications.

*Dept. of Computer Science, Ben Gurion University of the Negev, Be'er Sheva, Israel.
†Dept. of Computer Science, Technion, Haifa, Israel.

1 Introduction

How bad are the worst functions? For most standard complexity measures of boolean functions, the answer to this question is well known. For instance, the circuit complexity of the worst function $f : [N] \to \{0,1\}$ is $\Theta(N/\log N)$ [56, 51] and the two-party communication complexity of the worst function $f : [N] \times [N] \to \{0,1\}$ is $\Theta(\log N)$ in every standard model of communication complexity [50].[1] In sharp contrast, this question is wide open for most natural complexity measures in information-theoretic cryptography that involve *communication* or *randomness* rather than computation. Standard counting techniques or information inequalities only yield very weak lower bounds, whereas the best known upper bounds are typically linear in the size of the input domain (and exponential in the bit-length of the inputs).

The only exceptions to this state of affairs are in the context of secure multiparty computation, where it is known that when a big majority of honest parties is guaranteed, the communication and randomness complexity can always be made sublinear in the input domain size [4, 41] (see Section 1.2 for discussion of these and other related works). However, no similar results were known for secure computation with no honest majority, and in particular in the two-party case.

In the present work we study the complexity of the worst-case functions in several standard models for information-theoretic secure two-party computation, along with a related problem in the area of secret sharing.

We restrict our attention to security against *passive* (aka semi-honest) adversaries. We will usually also restrict the attention to deterministic two-party functionalities captured by boolean functions $f : [N] \times [N] \to \{0,1\}$, where the output is learned by both parties.[2] In the following, the term "secure" will refer by default to perfect security in the context of positive results and to statistical security in the case of negative results. In this setting, we consider the following questions.

OT COMPLEXITY. The first model we consider is secure two-party computation in the *OT-hybrid model*, namely in a model where an ideal oracle implementing 1-out-of-2 oblivious transfer [54, 29] (of bits) is available. Secure computation in this model is motivated by the possibility of realizing OT using noisy communication channels [22], the equivalence between OT and a large class of complete functionalities [47, 48], and the possibility of efficiently precomputing [3] and (in the computational setting) extending [2, 37] OTs. See [44] for additional motivating discussion.

Viewing OT as an "atomic currency" for secure two-party computation, it is natural to study the minimal number of OT calls required for securely computing a given two-party functionality f. We refer to this quantity as the *OT complexity* of f. Special cases of this question were studied in several previous works (e.g., [25, 2, 60]), and a more systematic study was conducted in [10, 53].

The GMW protocol [32, 33, 31] shows that the OT complexity of any function f is at most twice the size of the smallest boolean circuit computing f. For most functions $f : [N] \times [N] \to \{0,1\}$, this only gives an upper bound of $O(N^2/\log N)$ on the OT complexity.[3]

A simpler and better upper bound can be obtained by using 1-out-of-N OT (denoted $\binom{N}{1}$-OT). Concretely, the first party P_1, on input x_1, prepares a truth-table of the function $f_{x_1}(x_2)$ obtained by restricting f to its own input, and using $\binom{N}{1}$-OT lets the second party P_2 select the entry of this table indexed by x_2. Since $\binom{N}{1}$-OT can be reduced to $N-1$ instances of standard OT [16], we get an upper bound of $N-1$ on the OT complexity of the worst-case f. This raises the following question:

Question 1.1. *What is the OT complexity of the worst function $f : [N] \times [N] \to \{0,1\}$? In particular, can every such f be securely realized using $o(N)$ OTs?*

Given the existence of constant-rate reductions between OT and any finite complete functionality [35, 43], the answer to Question 1.1 remains the same, up to a constant multiplicative factor, even if the OT oracle is replaced by a different complete functionality, such as binary symmetric channel. In particular, the OT complexity of f is asymptotically the same as the "AND complexity" of f considered in [10].

We will also be interested in a sampling variant of Question 1.1, where the goal is to securely sample from some probability distribution over output pairs from $[N] \times [N]$ using a minimal number of OTs. This

[1] Here and in the following, we let $[N]$ denote the set $\{1, 2, \ldots, N\}$ and naturally identify an input $x \in [N]$ with a $\lceil \log_2 N \rceil$-bit binary representation.

[2] Using standard reductions (cf. [31]), our results can be extended to general (possibly randomized or even reactive) functionalities that may deliver different outputs to the two parties. While some of our results can also be extended to the case of k-party secure computation, we focus here on the two-party case for simplicity.

[3] The GMW protocol can handle XOR and NOT gates for free, but it is not clear if this can be used to significantly lower the complexity of the worst-case functions. A negative result in a restricted computation model is given in [19].

captures the rate of securely reducing complex correlations to simple ones, a question which was recently studied in [53].

CORRELATED RANDOMNESS COMPLEXITY. The second model we consider is that of secure two-party computation with an arbitrary source of correlated randomness. That is, during an offline phase, which takes place before the inputs are known, the two parties are given a pair of random strings (r_1, r_2) drawn from some fixed joint distribution, where r_i is known only to P_i. During the online phase, once the inputs (x_1, x_2) are known, the parties can use their correlated random inputs, possibly together with independent secret coins, to securely evaluate f. This model can be viewed as a relaxation of the OT-hybrid model discussed above, since each OT call is easy to realize given correlated randomness corresponding to a random instance of OT [3]. The model is motivated by the possibility of generating the correlated randomness using semi-trusted servers or a secure interactive protocol, thus capturing the goal of minimizing the online complexity of secure computation via offline preprocessing. See [13, 42, 24] for additional discussion.

General correlations have several known advantages over OT correlations in the context of secure computation. Most relevant to our work is a result from [42], showing that for any $f : [N] \times [N] \to \{0, 1\}$ there is a source of correlated randomness (r_1, r_2) given which f can be realized using only $O(\log N)$ bits of communication. However, the *correlated randomness complexity* of this protocol, namely length of the random strings r_1, r_2, is $O(N^2)$. Minimizing the correlated randomness complexity is desirable because the correlated randomness needs to be communicated and stored until the online phase begins. The simple OT complexity upper bound discussed above also implies an $O(N)$ upper bound on the correlated randomness complexity of the worst functions. No better bound is known. This raises the following question:

Question 1.2. *What is the correlated randomness complexity of the worst function $f : [N] \times [N] \to \{0, 1\}$? In particular, can every such f be securely realized using $o(N)$ bits of correlated randomness?*

COMMUNICATION COMPLEXITY OF PRIVATE SIMULTANEOUS MESSAGES PROTOCOLS. Feige, Kilian, and Naor [30] considered the following non-interactive model for secure two-party computation. The two parties simultaneously send messages to an external referee, where the message of party P_i depends on its input x_i and a common source of randomness r that is kept secret from the referee. From the two messages it receives, the referee should be able to recover $f(x_1, x_2)$ but learn no additional information about x_1, x_2. Following [38], we refer to such a protocol as a *private simultaneous messages* (PSM) protocol for f. A PSM protocol for f can be alternatively viewed as a special type of randomized encoding of f [39, 1], where the output of f is encoded by the output of a randomized function $\hat{f}((x_1, x_2); r)$ such that $\hat{f}$ can be written as $\hat{f}((x_1, x_2); r) = (\hat{f}_1(x_1; r), \hat{f}_2(x_2; r))$. This is referred to as a "2-decomposable" encoding in [36].

It was shown in [30] that every $f : [N] \times [N] \to \{0, 1\}$ admits a PSM protocol with $O(N)$ bits of communication. While better protocols are known for functions that have small formulas or branching programs [30, 38], this still remains the best known upper bound on the communication complexity of the worst-case functions, or even most functions, in this model. We thus ask:

Question 1.3. *What is the PSM communication complexity of the worst function $f : [N] \times [N] \to \{0, 1\}$? In particular, does every such f admit a PSM protocol which uses $o(N)$ bits of communication?*

SHARE COMPLEXITY OF FORBIDDEN GRAPH ACCESS STRUCTURES. A longstanding open question in information-theoretic cryptography is whether every (monotone) access structures can be realized by a secret-sharing scheme in which the share size of each party is polynomial in the number of parties. Here we consider a "scaled down" version of this question, where the access structure only specifies, for each *pair* of parties, whether this pair should be able to reconstruct the secret from its joint shares or learn nothing about the secret.[4] This type of graph-based access structures was considered in [58] under the name "forbidden graph" access structures.

A simple way of realizing such an access structure is by independently sharing the secret between each authorized pair. For most graphs, this solution distributes a share of length $\Omega(N)$ to each party. This can be improved by using covers by complete bipartite graphs implying that every graph access structure can be realized by a scheme in which the share size of each party is $O(N/\log N)$ [17, 15, 28]. This raises the following question:

[4] In contrast to the more standard notion of graph-based access structures, we make no explicit requirement on bigger or smaller sets of parties. However, one can easily enforce the requirement that every single party learns nothing about the secret and every set of 3 parties can reconstruct the secret.

Question 1.4. *What is share length required for realizing the worst graphs G? In particular, can every forbidden graph access structure on N nodes be realized by a secret-sharing scheme in which each party receives $o(N/\log N)$ bits?*

1.1 Our Results

For each of the above questions, we obtain the first $o(N/\log N)$ complexity upper bound. Our upper bounds are obtained by applying general transformations to variants of information-theoretic private information retrieval (PIR) protocols from the literature (see Section 1.2), where different flavors of PIR are required for different applications. At a high level, our results exploit new connections between 2-server PIR and OT complexity, between 3-server PIR and correlated randomness complexity, and between a special "decomposable" variant of 3-server PIR and PSM complexity. The secret sharing result is obtained by applying a general transformation to the PSM result, in the spirit of a transformation implicit in [8].

More concretely, we obtain the following main results.

OT COMPLEXITY OF SECURE TWO-PARTY COMPUTATION. We show that every function $f : [N] \times [N] \to \{0,1\}$ can be securely evaluated using $\widetilde{O}(N^{2/3})$ invocations of an oblivious transfer oracle. In fact, the total communication complexity and randomness complexity of the protocol are also bounded by $\widetilde{O}(N^{2/3})$. We also obtain a similar result for securely sampling a uniform pair of outputs from a set $S \subseteq [N] \times [N]$. More generally and precisely, for any joint probability distribution (U, V) over $[N] \times [N]$ and any $\epsilon > 0$, we obtain an ϵ-secure protocol for sampling correlated outputs from (U, V) using $N^{2/3} \cdot \mathrm{poly}(\log N, \log 1/\epsilon)$ OTs. This can be viewed as a nontrivial secure reduction of complex correlations (or "channels") to simple ones. These results apply the 2-server PIR protocol from [20].

CORRELATED RANDOMNESS COMPLEXITY OF SECURE TWO-PARTY COMPUTATION. We show that every function $f : [N] \times [N] \to \{0,1\}$ can be securely evaluated using $2^{\widetilde{O}(\sqrt{\log N})}$ bits of correlated randomness. In fact, the same bound holds also for the total randomness complexity of the protocol (counting private independent coins as well) and also for the *communication* complexity of the protocol. This result applies the 3-server PIR protocol of [27]. It was previously observed in [30, 42] that secure two-party computation with correlated randomness gives rise to a 3-server PIR protocol. Here we show a connection in the other direction.

COMMUNICATION COMPLEXITY OF PRIVATE SIMULTANEOUS MESSAGES. We show that every function $f : [N] \times [N] \to \{0,1\}$ can be realized by a PSM protocol with messages of length $O(\sqrt{N})$. The construction is based on a special "decomposable" variant of 3-server PIR which we realize by modifying a PIR protocol from [20]. In the hope of improving the $O(\sqrt{N})$ upper bound, we reduce the problem of decomposable 3-server PIR to a combinatorial question of obtaining a decomposable variant of matching vector sets [61, 26]. We leave open the existence of decomposable matching vector sets with good parameters.

In the terminology of randomized encoding of functions, the above result shows that every $f : [N] \times [N] \to \{0,1\}$ admits a 2-decomposable randomized encoding of length $O(\sqrt{N})$. It is instructive to note that whereas previous PSM protocols from [30, 38] employ a *universal* decoder (i.e., referee algorithm), which does not depend on the function f other than on a size parameter, the decoder in our construction strongly depends on f. It follows by a simple counting argument that this is inherent.

SHARE COMPLEXITY OF FORBIDDEN GRAPH ACCESS STRUCTURES. We show that for every graph G with N nodes, the corresponding forbidden graph access structure can be realized by a secret-sharing scheme in which each party gets a share of size $\widetilde{O}(\sqrt{N})$. This result is obtained by applying a general transformation to our new PSM protocols. Curiously, while our secret-sharing scheme is not linear, a simple generalization of a result of Mintz [52] implies a lower bound of $\Omega(\sqrt{N})$ on the share complexity of any *linear* scheme realizing the worst forbidden graph access structure. This extends a previous lower bound from [5] that applies to the stricter notion of graph-based access structures. The existence of *linear* secret-sharing schemes meeting this lower bound is left open.

1.2 Related Work

Prior to our work, the only previous context in which sublinear communication that of secure multiparty computation in the presence of an honest majority. While the complexity of standard protocols [11, 18] grows linearly with the circuit size, it is possible to do much better when there is a sufficiently large majority of honest parties. Beaver et al. [4] have shown that when only $\log n$ parties are corrupted, any function $f : \{0,1\}^n \to \{0,1\}$ can be securely evaluated using only $\mathrm{poly}(n)$ bits of communication and randomness, namely the complexity is polylogarithmic in the input domain size. Their technique makes an ad-hoc use of locally random reductions, which are in turn related to the problem of information-theoretic *private information retrieval* (PIR) [20]. A k-server PIR protocol allows a client to retrieve an arbitrary bit D_i from a database $D \in \{0,1\}^N$, which is held by k servers, while hiding the selection i from each individual server. The main optimization goal for PIR protocols is their communication complexity, which is required to be sublinear in N.

Ishai and Kushilevitz [41] present a general method for transforming communication-efficient PIR protocols into communication-efficient secure multiparty protocols in which the number of parties is independent of the total input length n. In contrast to our constructions, which require the underlying PIR protocols to satisfy additional computational and structural requirements, the transformation from [41] is completely general. On the down side, it does not apply in the two-party case and it requires PIR protocols with polylogarithmic communication, which are not known to exist for a constant number of servers k.

Beimel and Malkin [10] put forward the general goal of studying the minimal number of OTs/ANDs required for securely realizing a given two-party functionality f, observe that this quantity can be smaller in some cases than the circuit size of f, and obtain several connections between this question and communication complexity. These connections are mainly useful for proving lower bounds that are logarithmic in the domain size N or upper bounds for specific functions that have low communication complexity. More results in this direction are given in [46]. Prabhakaran and Prabhakaran [53] put forward the question of characterizing the rate of secure reductions between *sampling* functionalities, and strengthen previous negative results from [60] on the rate of secure reductions between different OT correlations. None of the above results give nontrivial upper bounds for the worst (or most) functions f. Winkler and Wulschlegger [60] prove an $\Omega(\log N)$ lower bound on the correlated randomness complexity of secure two-party computation. Except for very few functions, this lower bound is very far from the best known upper bounds even when considering the results of this work.

The complexity of secret sharing for graph-based access structures was extensively studied in a setting where the edges of the graph represent the *only* minimal authorized sets, that is, any set of parties that does not contain an edge should learn nothing about the secret. The notion of forbidden graph access structures we study, originally introduced in [58], can be viewed as a natural "promise version" of this question, where one is only concerned about sets of size 2.

It is known that every graph access structure can be realized by a scheme in which the share size of each party is $O(N/\log N)$ [17, 15, 28]. (This scheme is linear.) The best lower bound for the total share size required to realize a graph access structure by a general secret-sharing scheme is $\Omega(N \log N)$ [59, 14, 23]. The best lower bound for total share size required to realize a graph access structure by a linear secret-sharing scheme is $\Omega(N^{3/2})$ [5]. The problem of secret sharing for dense graphs was studied in [7]. More works on secret sharing of graph access structures can be found in [7].

2 Preliminaries

2.1 Models and Definitions

Notation. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $\mathcal{F}_N$ denote the set of all boolean functions from $[N] \times [N]$ to {0,1}. We will interpret $f \in \mathcal{F}_N$ as a 2-party function from $[N] \times [N]$ to {0,1}.

For an algorithm $\mathcal{B}$, let $\tau(\mathcal{B})$ denote the size of a boolean circuit (measured as the number of AND gates, in particular ignoring the number of XOR and NOT gates) that represents $\mathcal{B}$. Formally,

Definition 2.1. *Let $\mathcal{B}$ be an arbitrary (randomized) algorithm that takes inputs of length ℓ and produces output of length ℓ'. A* boolean circuit realizing $\mathcal{B}$ over basis $\{\wedge, \oplus, \neg\}$*, denoted by $C(\mathcal{B})$ is an ℓ-input, ℓ'-output boolean circuit over the basis $\{\wedge, \oplus, \neg\}$ (i.e., the logical operations AND, XOR, and NOT) such that the input-output behavior of $C(\mathcal{B})$ is identical to the input-output behavior of $\mathcal{B}$. Define $\tau(C(\mathcal{B}))$ as the number of AND gates (ignoring XOR and NOT gates) in the circuit $C(\mathcal{B})$. Define $\tau(\mathcal{B})$ as $\min_{C(\mathcal{B})} \tau(C(\mathcal{B}))$.*

Computational model. Since our results refer to *perfect* security, we incorporate perfect uniform sampling of $[m]$, for an arbitrary positive integer m, into the computational model as an atomic computation step.

Protocols. A k-party protocol can be formally defined by a *next message function*. This function on input (i, x_i, j, m) specifies a k-tuple of messages sent by party P_i in round j, when x_i is its input and m describes all the messages P_i received in previous rounds. The next message function may also instruct P_i to terminate the protocol, in which case it also specifies the output of P_i.

Protocols with preprocessing. In the *preprocessing model*, the specification of a protocol also includes a joint distribution $\mathcal{D}$ over $R_1 \times R_2 \ldots \times R_k$, where the R_i's are finite randomness domains. This distribution is used for sampling correlated random inputs $(r_1, \ldots, r_k)$ that the parties receive before the beginning of the protocol (in particular, the preprocessing is independent of the inputs). The next message function, in this case, may also depend on the private random input r_i received by P_i from $\mathcal{D}$. We assume that for every possible choice of inputs and random inputs, all parties eventually terminate.

OT correlations and the OT-hybrid model. We will be interested in the special case of the 2-party setting when the correlated random inputs (X, Y) given to the two parties are random OT correlations, corresponding to a random instance of oblivious transfer, in which the receiver obtains one of two bits held by the sender. That is, $X = (X_0, X_1)$ is uniformly random over $\{0,1\}^2$ and $Y = (b, X_b)$ for a random bit b. We refer to a model in which the correlated randomness given to the parties consists of random OT correlations, as the *OT preprocessing model*. Alternatively, we may consider a setting where (each pair of) parties have access to an ideal (bit) OT functionality that receives from one of the parties, designated as the sender, a pair of bits (x_0, x_1), and a choice bit b from the other party, designated as the receiver, and sends back to the receiver the value x_b. We call this model the *OT-hybrid model*.

Security definition. We use the standard ideal-world/real-world simulation paradigm. We restrict our attention to the case of semi-honest (passive) corruptions. Using the standard terminology of secure computation, the preprocessing model can be thought of as a *hybrid model* where the parties have a one-time access to an ideal randomized functionality $\mathcal{D}$ (with no inputs) providing them with correlated, private random inputs r_i. For lack of space, we omit the full security definitions (see, e.g., [42, App. A] adapted to the semi-honest setting).

2.2 Private Information Retrieval

The following is a somewhat non-standard view of PIR protocols, where the index is thought of as a pointer into a two-dimensional table, which in turn is thought of as a two-argument function.

Definition 2.2 (Private Information Retrieval)**.** *Let $\mathcal{F}_N$ be the set of all boolean functions $f : [N] \times [N] \to \{0,1\}$. A k-server private information retrieval (PIR) scheme $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ for $\mathcal{F}_N$ is composed of three algorithms: a randomized query algorithm $\mathcal{Q}$, an answering algorithm $\mathcal{A}$, and a reconstruction algorithm $\mathcal{R}$. At the beginning of the protocol, the client has an input $x \in [N] \times [N]$ and each server has an identical input f representing a function in $\mathcal{F}_N$. Using its private randomness $r \in \{0,1\}^{\gamma(N)}$, the client computes a tuple of k queries $(q_1, \ldots, q_k) = \mathcal{Q}(x, r)$, where $q_i \in \{0,1\}^{\alpha(N)}$, for all $i \in [k]$. The client then sends the query q_j to server $\mathcal{S}_j$, for every $j \in [k]$. Each server $\mathcal{S}_j$ responds with an answer $a_j = \mathcal{A}(j, q_j, f)$, with $a_j \in \{0,1\}^{\beta(N)}$. Finally, the client computes the value $f(x)$ by applying the reconstruction algorithm $\mathcal{R}(x, r, a_1, \ldots, a_k)$. We ask for the following correctness and privacy requirements:*

Correctness. *The client always outputs the correct value of $f(x)$. Formally, for every function $f \in \mathcal{F}_N$, every input $x \in [N] \times [N]$, and every random string r, if $(q_1, \ldots, q_k) = \mathcal{Q}(x, r)$ and $a_j = \mathcal{A}(j, q_j, f)$, for $j = 1, \ldots, k$, then $\mathcal{R}(x, r, a_1, \ldots, a_k) = f(x)$.*

Client's privacy. *Each server learns no information about x. Formally, for every two inputs $x, x' \in [N] \times [N]$, every $j \in [k]$, and every query q, the server $\mathcal{S}_j$ cannot know if the query was generated with input x or with input x'; that is, $\Pr[\mathcal{Q}_j(x, r) = q] = \Pr[\mathcal{Q}_j(x', r) = q]$, where $\mathcal{Q}_j$ denotes the jth query in the k-tuple that $\mathcal{Q}$ outputs and the probability is taken over a uniform choice of the random string r.*

The communication complexity *of a protocol $\mathcal{P}$ is the total number of bits communicated between the client and the k servers (i.e., $\sum_j (|q_j| + |a_j|) = k(\alpha(N) + \beta(N))$), maximized over the choice of $x \in [N] \times [N]$, $f \in \mathcal{F}_N$, and the random string $r \in \{0,1\}^{\gamma(N)}$.*

Every function $f \in \mathcal{F}_N$ is represented by an N^2-bit string $y = (y_{1,1}, \ldots, y_{N,N})$, where $f(i, j) = y_{i,j}$. The string y is also called a database, and we think of the client as querying a bit $y_{i,j}$ from the database.

Observe that the query received by each server is independent of the client's input x. In particular, this holds for the first query q_1, which therefore, may be thought of as depending only on the private randomness, say r, of the client, and not on the client input x. That is, we may assume that the query generation algorithm $\mathcal{Q}$ is expressed as the combination of two algorithms $\mathcal{Q}_1, \mathcal{Q}_{-1}$ such that query q_1 is generated by $\mathcal{Q}_1(r)$ while the remaining queries $q_2, \ldots, q_k$ are generated by algorithm $\mathcal{Q}_{-1}(x, r)$. In other words, we assume that the client with private randomness r, computes a tuple of k queries $(q_1, \ldots, q_k)$ as $q_1 = \mathcal{Q}_1(r)$, and $q_2, \ldots, q_k = \mathcal{Q}_{-1}(x, r)$.

2.3 Private Simultaneous Messages

The Private Simultaneous Messages (PSM) model was introduced by [30] as a minimal model for secure computation. It allows k players $P_1, \ldots, P_k$ with access to shared randomness, to send a single message each to a referee Ref, so that the referee learns the value of a function $f(x_1, \ldots, x_k)$ (where x_i is the private input of P_i) but nothing else. It is formally defined as follows:

Definition 2.3 (Private Simultaneous Messages)**.** *Let $X_1, \ldots, X_k, Z$ be finite domains, and let $X = X_1 \times \cdots \times X_k$. A private simultaneous messages (PSM) protocol $\mathcal{P}$, computing a k-argument function $f : X \to Z$, consists of:*

- *A finite domain R of shared random inputs, and k finite message domains $M_1, \ldots, M_k$.*
- *Message computation function $\mu_1, \ldots, \mu_k$, where $\mu_i : X_i \times R \to M_i$.*
- *A reconstruction function $g : M_1 \times \cdots \times M_k \to Z$.*

Let $\mu(x, r)$ denote the k-tuple of messages $(\mu_1(x_1, r), \ldots, \mu_k(x_k, r))$. We say that the protocol $\mathcal{P}$ is correct *(with respect to f), if for every input $x \in X$ and every random input $r \in R$, $g(\mu(x, r)) = f(x)$. We say that the protocol $\mathcal{P}$ is* private *(with respect to f), if the distribution of $\mu(x, r)$, where r is a uniformly random element of R, depends only on $f(x)$. That is, for every two inputs $x, x' \in X$ such that $f(x) = f(x')$, the random variables $\mu(x, r)$ and $\mu(x', r)$ (over a uniform choice of $r \in R$) are identically distributed. $\mathcal{P}$ is a PSM protocol computing f if it is both correct and private.*

The communication complexity of the PSM protocol $\mathcal{P}$ is naturally defined as $\sum_{i=1}^{n} \log |M_i|$. The randomness complexity of the PSM protocol $\mathcal{P}$ is defined as $\log |R|$.

Appendix A includes some additional definitions related to secret sharing.

3 Our Results

Secure computation in the OT-hybrid model. We show a connection between secure computation in the (bit) OT-hybrid model and 2-server PIR. More formally, we show:

Theorem 3.1. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 2-server PIR scheme for $\mathcal{F}_N$ as described in Definition 2.2. Let $\tau(\cdot)$ be as in Definition 2.1. Then, for any 2-party functionality $f : [N] \times [N] \to \{0, 1\}$, there is a protocol π which realizes f in the (bit) OT-hybrid model, and has the following features:*

- *π is perfectly secure against semi-honest parties;*
- *The total communication complexity, and in particular the number of calls to the OT oracle is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$.*

Plugging in parameters from the best known 2-server PIR protocol [20] (cf. Appendix B) in Theorem 3.1, we obtain:

Corollary 3.2. *For any 2-party functionality $f : [N] \times [N] \to \{0, 1\}$, there is a protocol π that realizes f in the (bit) OT-hybrid model; this protocol is perfectly secure against semi-honest parties, and has total communication complexity (including communication with the OT oracle) $\widetilde{O}(N^{2/3})$.*

Prior to our work, the best upper bound on the communication complexity of an information-theoretically secure protocol in the OT-hybrid model for evaluating arbitrary functions $f : [N] \times [N] \to \{0, 1\}$ was $\Omega(N)$. This can, for instance, be achieved by formulating the secure evaluation of $f : [N] \times [N] \to \{0, 1\}$ as a 1-out-of-N OT problem between the two parties, where party P_1 participates as sender with inputs $\{f(x_1, i)\}_{i \in [N]}$ and party P_2 participates as receiver with input x_2. An instance of 1-out-of-N OT can be obtained information theoretically from $O(N)$ instances of 1-out-of-2 OT by means of standard reductions [16].

Secure computation in the preprocessing model. Since OTs can be precomputed [3], the protocol implied by Theorem 3.1 yields a perfectly secure semi-honest protocol in the OT-preprocessing model where the communication complexity of the protocol and number of OTs required are both $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$.

Our next theorem shows that it is possible to obtain much better communication complexity in a setting where we are not restricted to using precomputed OT correlations alone. We show this by demonstrating a connection between secure computation in the preprocessing model and 3-server PIR. More formally,

Theorem 3.3. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 3-server PIR scheme for $\mathcal{F}_N$ as described in Definition 2.2. Let $\tau(\cdot)$ be as in Definition 2.1. Then, for any 2-party functionality $f : [N] \times [N] \to \{0,1\}$, there is a protocol π that realizes f in the preprocessing model, and has the following features:*

- *π is perfectly secure against semi-honest parties;*
- *The total communication complexity is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$;*
- *The total correlated randomness complexity is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$.*

Remark 3.4. We point out that a transformation in the other direction (i.e., constructing 3-server PIR protocols from protocols in the preprocessing model) was shown in [42]. In more detail, they show that a semi-honest secure protocol in the preprocessing model for $f : [N] \times [N] \to \{0,1\}$ with correlated randomness complexity $s(N)$ implies the existence of a 3-server, interactive PIR protocol, with communication complexity $s(\widehat{N}^{1/2}) + O(\log \widehat{N})$, where $\widehat{N}$ is the size of the database held by the servers. Taken together with our Theorem 3.3, this shows a two-way connection between the communication complexity of 3-server PIR protocols and the correlated randomness complexity of protocols in the preprocessing model.

Plugging in parameters from the best known 3-server PIR protocols [27, 9] (cf. Appendix C) in Theorem 3.3, we obtain:

Corollary 3.5. *For any 2-party functionality $f : [N] \times [N] \to \{0,1\}$, there is a protocol π that realizes f in the preprocessing model; this protocol is perfectly secure against semi-honest parties, and has total communication complexity and correlated randomness complexity $2^{\widetilde{O}(\sqrt{\log N})}$.*

The protocols implied by Corollaries 3.2 and 3.5 can be realized using 4 rounds of interaction (while preserving their communication and correlated randomness complexity). We point out that it is possible to obtain 2-round perfectly secure protocols in the preprocessing model but known constructions [16, 42] require correlated randomness complexity at least $O(N)$. In Appendix D, we show an upper bound of $O(N^{1/2})$ on the communication and correlated randomness complexity of 3-round perfectly secure protocols in the preprocessing model against semi-honest parties.

Private simultaneous messages (PSM) model. We obtain the following upper bound for 2-party protocols in the PSM model.

Theorem 3.6. *For any 2-party functionality $f : [N] \times [N] \to \{0,1\}$, there is a PSM protocol π that realizes f, and has the following features:*

- *π is perfectly secure against semi-honest parties;*
- *The total communication complexity and the complexity are $O(N^{1/2})$.*

This improves upon the best known upper bound of $O(N)$ on the communication and randomness complexity of PSM protocols [30].

Secret sharing for forbidden graph access structures. Consider a graph $G = (V, E)$. We are interested in the following graph access structure $\mathcal{A}^G$ in which the parties correspond to the vertices of the graph and (1) every vertex set of size three or more is authorized, and (2) every pair of vertices that is not connected by an edge in E is authorized. Such an access structure is called a *forbidden graph* access structure [58] since pairs of vertices connected by an edge in G are forbidden from reconstructing the secret. We obtain the following upper bound on the share size for a secret-sharing scheme realizing $\mathcal{A}^G$, for all G.

Theorem 3.7. *Let $G = (V, E)$ be a graph with $|V| = N$, and let $\mathcal{A}^G$ be the corresponding access structure. Then, there exists a perfect secret-sharing scheme realizing $\mathcal{A}^G$ with total share size $O(N^{3/2} \log N)$.*

Preliminaries: Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 2-server PIR protocol where servers hold as database the truth table of a function $f : [N] \times [N] \to \{0,1\}$. Parties P_1, P_2 have inputs $x_1, x_2 \in [N]$ respectively. At the end of the protocol, both parties learn $z = f(x_1, x_2)$.

Protocol:

1. P_1, P_2 choose uniformly random $r^{(1)}, r^{(2)} \in \{0,1\}^{\gamma(N)}$, respectively (where $\gamma(N)$ is the size of the randomness required by algorithm $\mathcal{Q}$). Let $\widetilde{\mathcal{Q}}$ denote an algorithm that takes as input $(x_1, r^{(1)}), (x_2, r^{(2)})$ and runs algorithm $\mathcal{Q}(x_1 \| x_2, r^{(1)} \oplus r^{(2)})$. Party P_1 with inputs $(x_1, r^{(1)})$ and P_2 with inputs $(x_2, r^{(2)})$ run a 2-party semi-honest secure GMW protocol in the OT-hybrid model to evaluate circuit $C(\widetilde{\mathcal{Q}})$. Let q_1, q_2 denote their respective outputs.
2. P_1 and P_2 locally compute $a_1 = \mathcal{A}(1, q_1, f)$ and $a_2 = \mathcal{A}(2, q_2, f)$ respectively.
3. Let $\widetilde{\mathcal{R}}$ denote an algorithm that takes as input $(a_1, x_1, r^{(1)}), (a_2, x_2, r^{(2)})$ and runs algorithm $\mathcal{R}(x_1 \| x_2, r^{(1)} \oplus r^{(2)}, a_1, a_2)$. Party P_1 with inputs $(a_1, x_1, r^{(1)})$ and P_2 with inputs $(a_2, x_2, r^{(2)})$ run a 2-party semi-honest secure GMW protocol in the OT-hybrid model to evaluate circuit $C(\widetilde{\mathcal{R}})$, where z denotes their common output. Both parties output z and terminate the protocol.

Figure 1: A perfectly secure protocol in the OT-hybrid model.

4 Secure computation in the OT-hybrid model

In this section, we construct a 2-party secure computation protocol realizing $f : [N] \times [N] \to \{0,1\}$ in the (bit) OT-hybrid model from a 2-server PIR protocol $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$. The resulting protocol will have communication complexity $\widetilde{O}(N^{2/3})$ and makes $\widetilde{O}(N^{2/3})$ calls to the ideal OT functionality, improving over prior work whose worst-case complexity (both in terms of communication and calls to the ideal OT functionality) was $\Omega(N)$ [16, 25].

Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 2-server PIR protocol. Let the truth table of the function $f : [N] \times [N] \to \{0,1\}$ that we are interested in, serve as the database (of length N^2). The high level idea behind our protocol is that the two parties P_1 and P_2, with their respective inputs x_1, x_2, securely emulate a virtual client with input $x = x_1 \| x_2$, and two virtual servers holding as database the truth table of f, in the PIR protocol $\mathcal{P}$. In more detail, parties P_1 and P_2, with inputs $x_1 \in [N], r^{(1)} \in \{0,1\}^{\gamma(N)}$ and $x_2 \in [N], r^{(2)} \in \{0,1\}^{\gamma(N)}$ respectively, emulate a PIR client by securely evaluating the query generation algorithm $\mathcal{Q}$ on input $x = x_1 \| x_2 \in [N^2]$ and randomness $r = r^{(1)} \oplus r^{(2)}$, such that party P_1 obtains query q_1 and party P_2 obtains query q_2. Then, using the PIR queries as their respective inputs, the parties locally emulate the PIR servers by running the PIR answer generation algorithm $\mathcal{A}$ and obtaining PIR answers a_1 and a_2, respectively. Finally, using the answers a_1, a_2, the inputs x_1, x_2, and the randomness $r^{(1)}$, $r^{(2)}$, parties P_1 and P_2 once again participate in a secure computation protocol to securely evaluate the PIR reconstruction algorithm $\mathcal{R}$ to obtain the final output z. The protocol is described in Figure 1. It is easy to see that the communication complexity as well as the number of calls to the ideal OT functionality is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$, that is, the complexity is proportional to the circuit size of the query and reconstruction algorithms. For a detailed proof, see Appendix K.1.

Intuitively, the protocol is private because (1) each individual PIR query does not leak any information about the query location and the reconstruction algorithms outputs nothing but the desired bit (both follow from the definition of PIR schemes); and (2) emulation of the algorithms run by the PIR client is done via secure computation protocols.

Instantiating the protocol in Figure 1 with the 2-server PIR protocol of Chor et al. [20] yields a perfectly secure protocol in the OT-hybrid model whose communication complexity is $\widetilde{O}(N^{2/3})$ and which makes $\widetilde{O}(N^{2/3})$ calls to the ideal OT functionality. See Appendix B, for a description of the 2-server PIR protocol and analysis of its efficiency. This proves Corollary 3.2.

5 Secure Computation in the Preprocessing Model

In this section, we construct a 2-party secure computation protocol realizing $f : [N] \times [N] \to \{0,1\}$ in the preprocessing model from a 3-server PIR protocol $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$. The resulting protocol will have communication and correlated randomness complexity $2^{\widetilde{O}(\sqrt{\log N})}$ improving over prior work whose worst-case complexity was $\Omega(N)$ [16, 25]. Note that we manage to emulate a protocol with 3 servers and one client by a protocol with 2 parties.

Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 3-server PIR protocol. We assume that the database represents the truth table of the function $f : [N] \times [N] \to \{0,1\}$ that we are interested in. The high level idea behind our protocol is that the two parties P_1 and P_2 with their respective inputs x_1, x_2 securely emulate a virtual client with input $x = x_1 \| x_2$, and two of the three virtual servers, say $\mathcal{S}_2$ and $\mathcal{S}_3$, holding as database the truth table of f, in the PIR protocol $\mathcal{P}$. The key observation is that server $\mathcal{S}_1$'s inputs and outputs can be precomputed and shared between P_1 and P_2 as preprocessed input. This is possible because $\mathcal{S}_1$'s input, namely the PIR query q_1, is distributed independently of the client's input, and thus can be computed beforehand. Similarly, a_1, the answer of $\mathcal{S}_1$, is completely determined by q_1 and the truth table of the function f, and thus can be precomputed as well. Thus, the preprocessed input along with the emulation done by P_1 and P_2 allow them to securely emulate all PIR algorithms $\mathcal{Q}$, $\mathcal{A}$, $\mathcal{R}$ of the 3-server PIR protocol $\mathcal{P}$. We provide a more detailed description of the protocol below.

Parties P_1 and P_2, are provided as preprocessed input, values ($r^{(1)}$,$a_1^{(1)}$) and ($r^{(2)}$, $a_1^{(2)}$) respectively along with sufficient OT correlations (whose use we will see later). The values $r^{(1)}$ and $r^{(2)}$ together determine the randomness used in PIR query generation algorithm $\mathcal{Q}$ as $r = r^{(1)} \oplus r^{(2)}$. Given randomness r, the first server's query q_1 (resp. answer a_1) is completely determined as $\mathcal{Q}_1(r)$ (resp. $\mathcal{A}(1, q_1, f)$). The values $a_1^{(1)}$ and $a_1^{(2)}$ together form a random additive sharing of a_1.

In the online phase, when parties obtain their respective inputs x_1 and x_2, they proceed to emulate the PIR client by securely evaluating the query generation algorithm on input $x = x_1 \| x_2 \in [N^2]$ and randomness $r = r^{(1)} \oplus r^{(2)}$, such that party P_1 obtains query q_2 and party P_2 obtains query q_3. Then, using the PIR queries as their respective inputs, the parties locally emulate the PIR servers by running the PIR answer generation algorithm $\mathcal{A}$ and obtain PIR answers a_2 and a_3 respectively. Recall that a random sharing of answer a_1 is already provided to the parties as preprocessed input. Using this random sharing of answer a_1, the locally computed answers a_2, a_3, the inputs x_1, x_2, and the randomness $r = r^{(1)} \oplus r^{(2)}$, parties P_1 and P_2 once again participate in a secure computation protocol to securely evaluate the PIR reconstruction algorithm $\mathcal{R}$ to obtain the final output z. The protocol is described in Figure 2. It is easy to see that the communication and correlated randomness complexity of the protocol equals $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$.

Intuitively, the protocol is private because (1) each party knows at most one PIR query, and (2) each individual PIR query does not leak any information about the query location (follows from the definition of PIR properties), and (3) emulation of the algorithms run by the PIR client is done via secure computation protocols. For a detailed proof, see Appendix K.2.

Instantiating the protocol in Figure 2 with the best known 3-server PIR protocol [61, 27, 9] we obtain a perfectly secure protocol in the preprocessing model whose communication and correlated randomness complexity is $2^{\widetilde{O}(\sqrt{\log N})}$. See Appendix C for a description of the concrete 3-server PIR protocol that we use for the instantiation, and also an analysis of efficiency of the protocol in Figure 2.

6 Private Simultaneous Messages

In this section, we provide a new framework for constructing PSM protocols (cf. Definition 2.3). Our proposed framework is based on a new variant of PIR protocols that we call *decomposable* PIR protocols. We define decomposable PIR in Section 6.1. We construct a 2-party PSM protocol using 3-server decomposable PIR protocols in Section 6.2, and we present a concrete decomposable 3-server PIR protocol in Section 6.3. The PSM protocol of Section 6.2, instantiated with this concrete decomposable 3-server PIR protocol, has communication (and randomness) complexity $O(N^{1/2})$, for all $f : [N] \times [N] \to \{0,1\}$.

6.1 Decomposable PIR Schemes

A k-server decomposable PIR protocol allows a client with input $x = (x_1, \ldots, x_{k-1}) \in [N]^{k-1}$ to query k servers, each holding a copy of a database of size N^{k-1} and retrieve the contents of the database at index x while offering (possibly relaxed) privacy guarantees to the client. Loosely speaking, decomposable PIR protocols differ from standard PIR protocols (cf. Definition 2.2) in two ways: (1) the query generation and reconstruction algorithms can be decomposed into "simpler" algorithms that depend only on parts of the entire input. (2) We change the privacy requirement and require that the query of server $\mathcal{S}_k$ together with some information about the answers of the first $k-1$ servers does not disclose information about the input of the client. We note that the privacy of the first $k-1$ queries follows from the decomposability of the query generation algorithm. We provide the formal definition below.

Definition 6.1 (Decomposable PIR)**.** *Let $\mathcal{F}_{N,k-1}$ be the set of all boolean functions $f : [N]^{k-1} \to \{0,1\}$. A k-server* decomposable PIR protocol $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ *for $\mathcal{F}_{N,k-1}$ consists of three algorithms: a randomized query algorithm $\mathcal{Q}$, an answering algorithm $\mathcal{A}$, and a reconstruction algorithm $\mathcal{R}$. At the beginning of the*

protocol, the client has an input $x = (x_1, \ldots, x_{k-1}) \in [N]^{k-1}$ (i.e., x is from the input domain of $\mathcal{F}_{N,k-1}$) and each server has an identical input f representing a function in $\mathcal{F}_{N,k-1}$. Using its private randomness $r \in \{0,1\}^{\gamma(N)}$, the client computes a tuple of k queries $(q_1, \ldots, q_k) = \mathcal{Q}(x, r)$, where each $q_i \in \{0,1\}^{\alpha(N)}$. The client then sends the query q_j to server $\mathcal{S}_j$, for every $j \in [k]$. Each server $\mathcal{S}_j$ responds with an answer $a_j = \mathcal{A}(j, q_j, f)$, with $a_j \in \{0,1\}^{\beta(N)}$. Finally, the client computes the value $f(x)$ by applying the reconstruction algorithm $\mathcal{R}(x, r, a_1, \ldots, a_k)$. The query generation algorithm $\mathcal{Q}$ and the reconstruction algorithm $\mathcal{R}$ satisfy the following "decomposability" properties.

Decomposable query generation algorithm. *The randomized query generation algorithm $\mathcal{Q}$ can be decomposed into k algorithms $\mathcal{Q}_1, \ldots, \mathcal{Q}_{k-1}, \mathcal{Q}_k = (\mathcal{Q}_k^1, \ldots, \mathcal{Q}_k^{k-1})$, such that for every input $x = (x_1, \ldots, x_{k-1}) \in [N]^{k-1}$, and for every random string $r \in \{0,1\}^{\gamma(N)}$, the queries $(q_1, \ldots, q_k) = \mathcal{Q}(x, r)$ are computed by the client as $q_j = \mathcal{Q}_j(x_j, r)$ for $j \in [k-1]$, and $q_k = (q_k^1, \ldots, q_k^{k-1}) = (\mathcal{Q}_k^1(x_1, r), \ldots, \mathcal{Q}_k^{k-1}(x_{k-1}, r))$.*

Decomposable reconstruction algorithm. *There exists algorithms $\mathcal{R}', \mathcal{R}''$ such that for every input $x = (x_1, \ldots, x_{k-1}) \in [N]^{k-1}$, and for every random string $r \in \{0,1\}^{\gamma(N)}$, if $(q_1, \ldots, q_k) = \mathcal{Q}(x, r)$, and $a_j = \mathcal{A}(j, q_j, f)$ for $j \in [k]$, then the output of the reconstruction algorithm $\mathcal{R}(x, r, a_1, \ldots, a_k)$ equals $\mathcal{R}''(a_k, \mathcal{R}'(x, r, a_1, \ldots, a_{k-1}))$.*

We ask for the following correctness and privacy requirements:

Correctness. *The client always outputs the correct value of $f(x)$. Formally, for every function $f \in \mathcal{F}_{N,k-1}$, every input $x \in [N]^{k-1}$, and every random string r, if $(q_1, \ldots, q_k) = \mathcal{Q}(x, r)$ and $a_j = \mathcal{A}(j, q_j, f)$, for $j \in [k]$, then $\mathcal{R}(x, r, a_1, \ldots, a_k) = f(x)$.*

Privacy. *We require that a_k, the answer of $\mathcal{S}_k$, and $\mathcal{R}'(x, r, a_1, \ldots, a_{k-1})$ do not disclose information not implied by $f(x)$. Formally, for every $f \in \mathcal{F}_{N,k-1}$, for every two inputs $x, x' \in [N]^{k-1}$ such that $f(x) = f(x')$, and every values q, b, letting $a_j = \mathcal{A}(j, \mathcal{Q}_j(x, r), f)$ and $a'_j = \mathcal{A}(j, \mathcal{Q}_j(x', r), f)$ for $j \in [k-1]$, and $q_k = \mathcal{Q}_k(x, r)$, $q'_k = \mathcal{Q}_k(x', r)$*

$$\Pr_r[q_k = q \wedge \mathcal{R}'(x, r, a_1, \ldots, a_{k-1}) = b] = \Pr_r[q'_k = q \wedge \mathcal{R}'(x', r, a'_1, \ldots, a'_{k-1}) = b],$$

where the probability is taken over a uniform choice of the random string r.

As usual, the communication complexity *of such a protocol $\mathcal{P}$ is the total number of bits communicated between the client and the k servers (i.e., $\sum_j (|q_j| + |a_j|) = k(\alpha(N) + \beta(N))$).*

6.2 From 3-Server Decomposable PIR to 2-Party PSM

Given a function $f : [N] \times [N] \to \{0,1\}$, we construct a 2-party PSM protocol for f using a 3-Server Decomposable PIR protocol. The protocol is formally described in Figure 3. We next give an informal description of the protocol. The shared randomness of the two parties is composed of two strings, one string for the decomposable PIR protocol and one for a PSM protocol for computing $\mathcal{R}'$. In the protocol, P_1, holding x_1, f, computes the query q_1 and its part of the query of server $\mathcal{S}_3$, namely q_3^1 (Party P_1 can compute these queries by the decomposability of the query generation). P_1 also computes a_1. Similarly, P_2, holding x_2, f, computes q_2, its part of the query of server $\mathcal{S}_3$, namely q_3^2, and a_2. Parties P_1 and P_2 send q_3^1 and q_3^2 to the referee, who uses this information and f to compute a_3. Furthermore, P_1 and P_2 execute a PSM protocol that enables the referee to compute $z' = \mathcal{R}'((x_1, x_2), r, a_1, a_2)$. The referee reconstructs $f(x)$ by computing $\mathcal{R}''(a_3, z')$, where a_3 is the answer computed by the referee for query $q_3 = (q_3^1, q_3^2)$.

The correctness of the protocol described in Figure 3 follows immediately from the definition of decomposable PIR. Furthermore, the information that the referee gets is q_3 and the messages of a PSM protocol computing $\mathcal{R}'$. By the privacy of the PSM protocol, the referee only learns the output of $\mathcal{R}'$ from this PSM protocol. Thus, the referee only learns q_3 and the output of $\mathcal{R}'$; by the privacy requirement of the decomposable PIR protocol the referee learns only $f(x)$. We summarize the properties of our PSM protocol in the following lemma.

Lemma 6.2. *Let $\mathcal{P}$ be a 3-server decomposable PIR protocol where the query length is $\alpha(N)$ and the randomness complexity is $\gamma(N)$. Furthermore, assume that $\mathcal{R}'$ can be computed by a 2-party PSM protocol with communication complexity $\alpha'(N)$ and randomness complexity $\gamma'(N)$. Then, every function $f \in \mathcal{F}_N$ can be computed by a 2-party PSM protocol with communication complexity $\alpha(N) + \alpha'(N)$ and randomness complexity $\gamma(N) + \gamma'(N)$.*

6.3 A 3-Server Decomposable PIR Protocol

In this section, we show how to construct a decomposable 3-server PIR protocol. Our construction is inspired by the cubes approach of [20]. We start with a high level description of this approach, specifically for the case of 4-dimensional cubes (see also the 3-dimensional case in Appendix B), and of its adaptation to the decomposable case. The formal description of the protocol is deferred to Appendix I.

The starting point of the CGKS cubes approach (restricted here to dimension 4) is viewing the n-bit database as a 4-dimensional cube (i.e., $[n^{1/4}]^4$). Correspondingly, the index that the client wishes to retrieve is viewed as a 4-tuple $i = (i_1, \ldots, i_4)$. The protocol starts by the client choosing a random subset for each dimension, i.e. $S_1, \ldots, S_4 \subseteq_R [n^{1/4}]$. It then creates 16 queries of the form $(T_1, \ldots, T_4)$ where each T_j is either S_j itself or $S_j \oplus \{i_j\}$ (we often use vectors in $\{0,1\}^4$ to describe these 16 combinations; e.g., 0000 refers to the query $(S_1, \ldots, S_4)$ while 1111 refers to the query $(S_1 \oplus \{x_1\}, \ldots, S_4 \oplus \{x_4\})$. If there were 16 servers available, the cliebt would send each query $(T_1, \ldots, T_4)$ to a different server ($4 \cdot n^{1/4}$ bits to each), who will reply with a single bit which is the XOR of all bits in the sub-cube $T_1 \otimes \ldots \otimes T_4$. The observation made in [20] is that each element of the cube appears in an even number of those 16 sub-cubes, and the only exception is the entry $i = (i_1, \ldots, i_4)$ that appears exactly once. Hence, taking the XOR of the 16 answer bits, all elements of the cube are canceled out except for the desired element in position i.

The next observation of the cubes approach is that a server who got a query $(T_1, \ldots, T_4)$ can provide a longer answer (but still of length $O(n^{1/4})$ bits) from which the answers to some of the other queries can be derived (and, hence, the corresponding servers in the initial solution can be eliminated). Specifically, it can provide also the answers to the queries $(T_1 \oplus \{\ell\}, T_2, T_3, T_4)$, for all possible values $\ell \in [n^{1/4}]$. One of these is the bit corresponding to $\ell = i_1$ which is the desired answer for another one of the 16 queries; and, clearly, the same can be repeated in each of the 4 dimensions. Stated in the terminology of 4-bit strings, a server that gets the query represented by some $b \in \{0,1\}^4$ can reply with $O(n^{1/4})$ bits from which the answer to the 5 queries of hamming distance at most one from b can be obtained; further, it can be seen that 4 servers that will answer the queries corresponding to $\{1100, 0011, 1000, 0111\}$ provide all the information to answer the 16 queries in the initial solution (this corresponds also to the notion of "covering codes" from the coding theory literature).

Next, we informally describe how to turn the above ideas into a decomposable 3-server PIR protocol. We still view the database as 4-dimensional cube and the client is still interested in obtaining the answers to the same 16 queries. Moreover, we are allowed to use only 3 servers for this. However, the requirements of decomposable PIR give us some freedom that we did not have before; specifically, we allow answer of the first server to depend on $x_1 = (i_1, i_2)$ and the answer of the second server to depend on $x_2 = (i_3, i_4)$. The query to the third server should still give no information about i. Specifically, we will give the first server the basic sets $S_1, \ldots, S_4$ along with the values i_1, i_2. This server can easily compute the answer to all 4 queries of the form $(T_1, \ldots, T_4)$ with T_1 being either S_1 or $S_1 \oplus \{i_1\}$ and T_2 being either S_2 or $S_2 \oplus \{i_2\}$ (in vectors notation, those correspond to the queries 0000,0100,1000,1100). Moreover, using the idea described above, even though the first server does not know the value of i_3 it can provide $O(n^{1/4})$-bit answer corresponding to all choices of i_3 from which the client can select the right ones (in vectors notation, those corresponding to the queries 0010,0110,1010,1110). Similarly it can provide $O(n^{1/4})$-bit answer corresponding to all choices of i_4 from which the client can select the right ones (in vectors notation, those corresponding to the queries 0001,0101,1001,1101). The query to the second server consists of $S_1, \ldots, S_4$ along with the values i_3, i_4. In a similar way, this server provides an answer of $O(n^{1/4})$ bits that can be used to answer the queries 0000,0010,0001,0011 directly and 1000,1010,1001,1011, 0100,0110,0101,0111 by enumerating all values of i_1 and then all values of i_2 (some queries are answered by both servers; this small overhead can be easily saved – see below). So, based on a_1, a_2, the only query that remained unanswered is the 1111 query. For this, the client asks the third server the query $(S_1 \oplus \{i_1\}, \ldots, S_4 \oplus \{i_4\})$ (which is independent of i) and gets the missing bit, denoted a_3, back. Finally, note that the reconstruction has the desired "decomposable" form: the client output can be obtained by processing the answers of the first two servers to get the sum v of the first 15 queries (this is the desired $\mathcal{R}'$) and then adding a_3 to it. Moreover, the pair (q_3, v) gives no information on i beyond the output: q_3 is independent of i (it is just a random sub-cube), and v is just the exclusive-or of the the output and a_3 (which depends only on q_3 and hence independent of i).

7 Secret Sharing

We present a generic transformation from any 2-party PSM protocol to secret-sharing schemes for forbidden graph access structures, and then use the results from Section 6 to obtain efficient secret-sharing schemes for these access structures. Specifically, we obtain N-party secret-sharing schemes for forbidden graph access

structures whose total share size is $O(N^{3/2})$. The best previous constructions for these access structures had total share size $O(N^2/\log N)$ [17, 15, 28].

In Section 7.1, we demonstrate our transformation from PSM protocols to secret-sharing schemes for forbidden graph access structures for the simple case when the graph is bipartite. In Section J, we generalize our construction to arbitrary graphs. We start by formally defining forbidden graph access structures.

Definition 7.1. *Let $G = (V, E)$ be an arbitrary graph. A* forbidden graph access structure*, denoted $\mathcal{A}^G$, is an access structure where the parties are the vertices in V and the only unauthorized sets are singletons (i.e., sets containing a single vertex in V), and sets of size 2 corresponding to edges on G (i.e., sets $\{x, y\}$ with $(x, y) \in E$).*

7.1 Secret Sharing Schemes for Forbidden Bipartite Graph Access Structures

We first show how to realize forbidden graph access structures $\mathcal{A}^G$, where the graph G is bipartite.

Definition 7.2. *Let $G = (L, R, E)$ be a bipartite graph, where $|L| = |R| = N$. We label the vertices in L by $1, 2, \ldots, N$, and similarly, vertices in R by $1, 2, \ldots, N$. We associate the bipartite graph $G = (L, R, E)$ with a boolean function $f_G : [N] \times [N] \to \{0, 1\}$, where $f(x, y)$ equals 0 iff there exists an edge between vertex $x \in L$ and vertex $y \in R$.*

Lemma 7.3. *Let $G = (L, R, E)$ be a bipartite graph where $|L| = |R| = N$ and $f_G : [N] \times [N] \to \{0, 1\}$ be the function associated with G. Let $\mathcal{P}$ be a PSM protocol for computing f_G with communication complexity $c_{\mathcal{P}}(N)$. Then, there exists a secret sharing realizing $\mathcal{A}^G$ with domain of secrets $\{0, 1\}$ and total share size $O(N \cdot c_{\mathcal{P}}(N))$.*

Proof. In a forbidden bipartite graph access structure the sets that can reconstruct the secret are: (1) All sets of 3 or more parties, (2) all pairs of parties that correspond to vertices from the same "side" of the graph (L or R), and (3) all pairs of parties that correspond to vertices from different sides of the graph and are not connected by an edge.

We construct a secret-sharing scheme for $\mathcal{A}^G$ by dealing with the three types of authorized sets. First, the dealer shares the secret with Shamir's 3-out-of-$2N$ threshold secret-sharing scheme among the $2N$ parties of the access structure. Next, the dealer independently shares the secret with Shamir's 2-out-of-N threshold secret-sharing scheme among the parties in L, and independently among the parties in R.

The interesting case is how to share the secret for sets $\{x, y\}$ such that $x \in L, y \in R$, and $(x, y) \notin E$. Let μ_1, μ_2 represent the message computation functions of the PSM protocol $\mathcal{P}$ (as defined in Definition 2.3). To share a secret $s \in \{0, 1\}$, the dealer chooses the randomness r, required for $\mathcal{P}$. Then, depending on the value of s, it distributes the shares to the parties as follows:

- If $s = 0$, then the dealer chooses arbitrary $x_0, y_0 \in [N]$ such that $f_G(x_0, y_0) = 0$, and gives the share $m_x = \mu_1(x_0, r)$ to each party $x \in L$, and the share $m_y = \mu_2(y_0, r)$ to each party $y \in R$.
- Else, if $s = 1$, then the dealer gives the share $m_x = \mu_1(x, r)$ to each party $x \in L$, and the share $m_y = \mu_2(y, r)$ to each party $y \in R$.

Any two parties $x \in L$ and $y \in R$ that are not connected by an edge in G reconstruct the secret by returning the output of the PSM reconstruction function $s' = g(m_x, m_y)$ (cf. Definition 2.3). Correctness of this reconstruction for $(x, y) \notin E$ follows from the correctness of the PSM protocol $\mathcal{P}$. Specifically, (1) when $s = 0$, the parties x and y reconstruct $f(x_0, y_0) = 0 = s$, and (2) when $s = 1$, the parties x and y reconstruct $f_G(x, y) = 1 = s$.

For the privacy, consider a pair of parties x, y such that $x \in L, y \in R$, and $(x, y) \in E$. When $s = 0$, these parties hold shares $\mu_1(x_0, r)$ and $\mu_2(y_0, r)$ respectively. When $s = 1$, these parties hold shares $\mu_1(x, r)$ and $\mu_2(y, r)$ respectively. Since $f_G(x, y) = f_G(x_0, y_0) = 0$, the shares do not reveal any information about s (by the privacy of the PSM protocol). □

Using the PSM protocols described in Theorem 3.6 in Lemma 7.3, we get the following corollary.

Corollary 7.4. *Let $G = (L, R, E)$ be a bipartite graph where $|L| = |R| = N$. There exists a secret sharing realizing $\mathcal{A}^G$ with domain of secrets $\{0, 1\}$ and total share size $O(N^{3/2})$.*

In Appendix J, we construct secret-sharing schemes realizing $\mathcal{A}^G$ for general graphs, using the secret-sharing scheme for forbidden bipartite graph access structures.

References

[1] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *CCC*, pages 260–274, 2005.

[2] D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *STOC*, pages 479–488, 1996.

[3] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *Advances in Cryptology — Crypto '95*, volume 963 of *LNCS*, pages 97–109. Springer, 1995.

[4] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology — Crypto '90*, volume 537 of *LNCS*, pages 62–76. Springer, 1991.

[5] A. Beimel, A. Gal, and M. Paterson. Lower bounds on monotone span programs. In *FOCS*, pages 674–681, 1995.

[6] Amos Beimel, Mike Burmester, Yvo Desmedt, and Eyal Kushilevitz. Computing functions of a shared secret. In *SIAM J. Discrete Math*, pages 324–345, 2000.

[7] Amos Beimel, Oriol Farras, and Yuval Mintz. Secret sharing for very dense graphs. In *Crypto*, pages 144–161, 2012.

[8] Amos Beimel and Yuval Ishai. On the power of nonlinear secret sharing. In *CCC*, pages 188–202, 2001.

[9] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *CCC*, pages 258–268, 2012.

[10] Amos Beimel and Tal Malkin. A quantitative approach to reductions in secure computation. In Moni Naor, editor, *1st Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 238–257. Springer, February 2004.

[11] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10. ACM Press, May 1988.

[12] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology — Crypto '88*, volume 403 of *LNCS*, pages 27–35. Springer, 1990.

[13] Rikke Bendlin, Ivan Damg*r*ard, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology — Eurocrypt 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, 2011.

[14] C. Blundo, A. De Santis, R. de Simone, and U. Vaccaro. Tight bounds on information rate of secret sharing schemes. In *Designs, Codes and Cryptography*, pages 107–122, 1997.

[15] C. Blundo, A. De Santis, L. Gargano, and U. Vaccaro. On information rate of secret sharing schemes. In *Theoretical Computer Science*, pages 283–306, 1996.

[16] G. Brassard, C. Crepeau, and J.-M. Robert. Information theoretic reduction among disclosure problems. In *FOCS*, pages 168–173, 1986.

[17] S. Bublitz. Decomposition of graphs and monotone formula size of homogeneous functions. In *Acta Informatica*, pages 689–696, 1986.

[18] David Chaum, Claude Crépeau, and Ivan Damg*r*ard. Multiparty unconditionally secure protocols. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19. ACM Press, May 1988.

[19] Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial derivatives in arithmetic complexity and beyond. In *Foundations and Trends in Theoretical Computer Science*, pages 1–138, 2011.

[20] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, pages 41–50, 1995.

[21] Richard Cleve. Towards optimal simulations of formulas by bounded-width programs. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 271–277, New York, NY, USA, 1990. ACM.

[22] C. Crepeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *FOCS*, pages 42–52, 1988.

[23] L. Csirmaz. Secret sharing schemes on graphs. In *ePrint 2005/059*, 2005.

[24] Ivan Damgard and Sarah Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC*, pages 621–641, 2013.

[25] Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In Mihir Bellare, editor, *Advances in Cryptology — Crypto 2000*, volume 1880 of *LNCS*, pages 74–92. Springer, 2000.

[26] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. In *51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 705–714. IEEE, 2010.

[27] Klim Efremenko. 3-query locally decodable codes of subexponential length. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 39–44. ACM Press, 2009.

[28] P. Erdos and L. Pyber. Covering a graph by complete bipartite graphs. In *Discrete Mathematics*, pages 249–251, 1997.

[29] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology — Crypto '82*, pages 205–210. Plenum Press, 1983.

[30] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 554–563. ACM Press, May 1994.

[31] Oded Goldreich. Foundations of cryptography - volume 2, basic applications. 2004.

[32] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229. ACM Press, May 1987.

[33] Oded Goldreich and Ronen Vainish. How to solve any protocol problem - an efficiency improvement. In Carl Pomerance, editor, *Advances in Cryptology — Crypto '87*, volume 293 of *LNCS*, pages 73–86. Springer, 1988.

[34] Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. In *COMBINATORICA (20)*, pages 71–86, 2000.

[35] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In Ran Canetti, editor, *5th Theory of Cryptography Conference — TCC 2008*, volume 4948 of *LNCS*, pages 393–411. Springer, March 2008.

[36] Yuval Ishai. Randomization techniques for secure computation. In *Secure Multi-Party Computation. Editors:* Manoj Prabhakaran and Amit Sahai, 2013.

[37] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology — Crypto 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, 2003.

[38] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, pages 174–184, 1997.

[39] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 294–304. IEEE, November 2000.

[40] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

[41] Yuval Ishai and Eyal Kushilevitz. On the hardness of information-theoretic multiparty computation. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of *LNCS*, pages 439–455. Springer, 2004.

[42] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, pages 600–620, 2013.

[43] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Extracting correlations. In *50th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 261–270. IEEE, 2009.

[44] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, 2008.

[45] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *GLOBECOM*, pages 99–102, 1987.

[46] Marc Kaplan, Iordanis Kerenidis, Sophie Laplante, and Jeremie Roland. Non-local box complexity and secure function evaluation. In *FSTTCS*, pages 239–250, 2009.

[47] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[48] Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 316–324. ACM Press, May 2000.

[49] Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In Bimal K. Roy, editor, *Advances in Cryptology — Asiacrypt 2005*, volume 3788 of *LNCS*, pages 136–155. Springer, December 2005.

[50] Eyal Kushilevitz and Noam Nisan. Communication complexity. 1997.

[51] O. B. Lupanov. A method of circuit synthesis. In *Izvesitya VUZ, Radiofizika*, pages 120–140, 1958.

[52] Yuval Mintz. Information ratios of graph secret-sharing schemes. Master's thesis, Ben Gurion University, Israel, 2012.

[53] Vinod Prabhakaran and Manoj Prabhakaran. Assisted common information with an application to secure two-party sampling. In *arxiv:1206.1282v1*, 2012.

[54] M. O. Rabin. How to exchange secrets with oblivious transfer. In *Technical Report TR-81, Aiken Computation Lab, Harvard University*, 1981.

[55] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC1. In *40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 554–567. IEEE, October 1999.

[56] C. Shannon. The synthesis of two-terminal switching circuits. In *Bell System Technical Journal*, pages 59–98, 1949.

[57] Douglas R. Stinson. Decomposition construction for secret sharing schemes. *IEEE Trans. on Information Theory*, 40(1):118–125, 1994.

[58] Hung-Min Sun and Shiuh-Pyng Shieh. Secret sharing in graph-based prohibited structures. In *INFOCOM*, pages 718–724, 1997.

[59] M. van Dijk. On the information rate of perfect secret sharing schemes. In *Designs, Codes and Cryptography*, pages 143–169, 1995.

[60] Severin Winkler and Jürg Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In *Advances in Cryptology — Crypto 2010*, volume 6223 of *LNCS*, pages 707–723. Springer, 2010.

[61] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 266–274. ACM Press, June 2007.

Preliminaries: Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 3-server PIR protocol where servers hold as database the truth table of a function $f : [N] \times [N] \to \{0,1\}$. Parties P_1, P_2 have inputs $x_1, x_2 \in [N]$ respectively. At the end of the protocol, both parties learn $z = f(x_1, x_2)$.

Preprocessing:

- Sample random $r^{(1)}, r^{(2)} \in \{0,1\}^{\gamma(N)}$, $a_1^{(1)} \leftarrow \{0,1\}^{\beta(N)}$.
- Compute $q_1 = \mathcal{Q}_1(r^{(1)} \oplus r^{(2)})$, and $a_1 = \mathcal{A}(1, q_1, f)$.
- Sample $(X', Y') = (X, Y)^{O(\tau(\mathcal{Q})+\tau(\mathcal{R}))}$, where (X, Y) represents a random OT correlation.
- Output $(r^{(1)}, a_1^{(1)}, X')$ to P_1, and $(r^{(2)}, a_1^{(2)} = a_1 \oplus a_1^{(1)}, Y')$ to P_2.

Protocol:

1. Let $\widetilde{\mathcal{Q}}$ denote an algorithm that takes as input $(x_1, r^{(1)}), (x_2, r^{(2)})$ and runs algorithm $\mathcal{Q}_{-1}(x_1 \| x_2, r^{(1)} \oplus r^{(2)})$. Party P_1 with inputs $(x_1, r^{(1)})$ and party P_2 with inputs $(x_2, r^{(2)})$ run a 2-party semi-honest secure GMW protocol, using random OT correlations of size $O(\tau(\mathcal{Q}))$ (derived from (X', Y')), to evaluate circuit $C(\widetilde{\mathcal{Q}})$. Let q_2, q_3 denote their respective outputs.
2. P_1 and P_2 locally compute $a_2 = \mathcal{A}(1, q_2, f)$ and $a_3 = \mathcal{A}(2, q_3, f)$ respectively.
3. Let $\widetilde{\mathcal{R}}$ denote an algorithm that takes as input $(a_1^{(1)}, a_2, x_1, r^{(1)}), (a_1^{(2)}, a_3, x_2, r^{(2)})$ and runs algorithm $\mathcal{R}(x_1 \| x_2, r^{(1)} \oplus r^{(2)}, a_1^{(1)} \oplus a_1^{(2)}, a_2, a_3)$. Party P_1 with inputs $(a_1^{(1)}, a_2, x_1, r^{(1)})$ and party P_2 with inputs $(a_1^{(2)}, a_3, x_2, r^{(2)})$ run a 2-party semi-honest secure GMW protocol, using random OT correlations of size $O(\tau(\mathcal{R}))$ (derived from (X', Y')), to evaluate circuit $C(\widetilde{\mathcal{R}})$. Let z denote their common output. Both parties output z and terminate the protocol.

Figure 2: A perfectly secure protocol in the preprocessing model.

A Additional Preliminaries

In this Appendix we include some additional preliminaries.

A.1 Secret Sharing

Definition A.1. *Let $P = \{P_1, \ldots, P_m\}$ be a set of parties. A collection $\Gamma \subseteq 2^P$ is* monotone *if $B \in \Gamma$ and $B \subseteq C$ imply that $C \in \Gamma$. An* access structure *is a monotone collection $\Gamma \subseteq 2^P$ of non-empty subsets of P. Sets in Γ are call* authorized*, and sets not in Γ are called* unauthorized*. The family of minimal authorized subsets in Γ is denoted by* $\min \Gamma$.

A distribution scheme $\Sigma = \langle \Pi, \mu \rangle$ with domain of secrets K is a pair, where μ is a probability distribution on some finite set R called the set of random strings and Π is a mapping from $K \times R$ to a set of m-tuples $K_1 \times K_2 \times \cdots \times K_m$, where K_j is called the domain of shares *of P_j. A dealer distributes a secret $k \in K$ according to Σ by first sampling a random string $r \in R$ according to μ, computing a vector of shares $\Pi(k, r) = (s_1, \ldots, s_m)$, and privately communicating each share s_j to party P_j. For a set $A \subseteq P$, we denote $\Pi(k, r)_A$ as the restriction of $\Pi(k, r)$ to its A-entries. The (normalized)* total share size *of a distribution scheme is $\sum_{j \in [m]} \log |K_j| / \log |K|$.*

Definition A.2 (Secret Sharing)**.** *Let K be a finite set of secrets, where $|K| \geq 2$. A distribution scheme $\langle \Pi, \mu \rangle$ with domain of secrets K is a* secret-sharing scheme *realizing an access structure Γ if the following two requirements hold:*

Correctness. *The secret k can be reconstructed by any authorized set of parties. That is, for any set $B = \{P_{i_1}, \ldots, P_{i_{|B|}}\} \in \Gamma$, there exists a reconstruction function $\mathrm{Recon}_B : K_{i_1} \times \cdots \times K_{i_{|B|}} \to K$ such that for every $k \in K$,*

$$\Pr_r[\mathrm{Recon}_B(\Pi(k, r)_B) = k] = 1.$$

Privacy. *Every unauthorized set cannot learn anything about the secret (in the information theoretic sense) from their shares. Formally, for any set $T \notin \Gamma$, for every two secrets $a, b \in K$, and for every possible vector of shares $\langle s_j \rangle_{P_j \in T}$,*

$$\Pr_r[\Pi(a, r)_T = \langle s_j \rangle_{P_j \in T}] \;=\; \Pr_r[\Pi(b, r)_T = \langle s_j \rangle_{P_j \in T}].$$

Preliminaries: Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 3-server decomposable PIR protocol (cf. Definition 6.1) where servers hold as database the truth table of a function $f : [N] \times [N] \to \{0,1\}$. As in Definition 6.1, we assume that (1) the query generation algorithm $\mathcal{Q}$ can be decomposed into algorithms $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3 = (\mathcal{Q}_3^1, \mathcal{Q}_3^2))$, and (2) the reconstruction algorithm $\mathcal{R}$ can be decomposed using algorithms $\mathcal{R}'$ and $\mathcal{R}''$.
Parties P_1, P_2 have inputs $x_1, x_2 \in [N]$ respectively. At the end of the protocol, the referee Ref learns $z = f(x_1, x_2)$.

Shared randomness: Both parties share uniform randomness:

- $r \in \{0,1\}^{\gamma(N)}$ required for the query generation algorithm $\mathcal{Q}$.
- $s \in \{0,1\}^*$ required for a 2-party PSM protocol $\pi_{\mathcal{R}'}$, where party P_1 holds (x_1, a_1) and party P_2 holds (x_2, a_2), and referee Ref obtains output $\mathcal{R}'((x_1, x_2), r, a_1, a_2)$.

Protocol:

1. Using its input x_1 and shared randomness r, party P_1 computes $q_1 = \mathcal{Q}_1(x_1, r)$, and $q_3^1 = \mathcal{Q}_3^1(x_1, r)$. It then computes $a_1 = \mathcal{A}(1, q_1, f)$. Using (x_1, a_1), and shared randomness s, party P_1 computes m_1 – its message to Ref in $\pi_{\mathcal{R}'}$. Finally, it sends (q_3^1, m_1) to Ref.

   Similarly, using its input x_2 and shared randomness r, party P_2 computes $q_2 = \mathcal{Q}_2(x_2, r)$, and $q_3^2 = \mathcal{Q}_3^2(x_2, r)$. It then computes $a_2 = \mathcal{A}(2, q_2, f)$. Using (x_2, a_2), and shared randomness s, party P_1 computes m_2 – its message to Ref in $\pi_{\mathcal{R}'}$. Finally, it sends (q_3^2, m_2) to Ref.
2. The referee Ref computes $a_3 = \mathcal{A}(3, q_3, f)$ where $q_3 = (q_3^1, q_3^2)$ (using the values received from P_1, P_2). Then, using the PSM messages m_1, m_2, referee Ref computes $z' = \mathcal{R}'((x_1, x_2), r, a_1, a_2)$, from which it computes its final output $z = \mathcal{R}''(a_3, z')$.

Figure 3: A 2-party perfectly secure PSM protocol.

Definition A.3 (CNF sharing [45])**.** *Let G be a finite Abelian group. The t-private CNF sharing over G is a generalization of additive sharing, where each party gets more than one group element. To share $s \in G$ among k parties, do the following:*

- *Additively share s into $\binom{k}{t}$ shares r_A, $A \in \binom{[k]}{t}$. The share of P_i consists of the $\binom{k-1}{t}$ group elements $\{r_A \ : \ i \notin A\}$.*

For each set $T \in \binom{[k]}{t}$, the parties in T do not get r_T and thus have no information about s. This implies that the scheme is t-private. We will be mostly interested in 1-private CNF sharing and refer to it simply as CNF sharing; in this case we will write r_i instead of $r_{\{i\}}$.

A.1.1 Share Conversion

Definition A.4 (Local share conversion [9])**.** *Let $\mathcal{L}$ and $\mathcal{L}'$ be two secret-sharing schemes over the domains of secrets K_1 and K_2, respectively, and let $C \subseteq K_1 \times K_2$ be a relation such that, for every $a \in K_1$, there exists at least one $b \in K_2$ such that $(a, b) \in C$. We say that $\mathcal{L}$ is* locally convertible *to $\mathcal{L}'$ with respect to C if there exist local conversion functions $g_1, \ldots, g_k$ such that*

- *If $s_1, \ldots, s_k$ is a valid sharing for some secret s in $\mathcal{L}$, then $g_1(s_1), \ldots, g_k(s_k)$ is a valid sharing for some secret s' in $\mathcal{L}'$ such that $(s, s') \in C$.*

Definition A.5 (The relation C_S [9])**.** *Let G and G' be finite Abelian groups and let $S \subseteq G \setminus \{0\}$. (We will sometimes view G and G' as the additive groups of rings and refer to rings instead of groups.) The relation C_S converts $s = 0 \in G$ to any* nonzero *$s' \in G'$ and every $s \in S$ to $s' = 0$. There is no requirement when $s \notin S \cup \{0\}$. Formally,*

$$C_S = \{(s,0) \ : \ s \in S\} \quad \cup \quad \{(0,s') \ : \ s' \in G' \setminus \{0\}\} \\ \cup \quad \{(s,s') \ : \ s \notin S \cup \{0\}, s' \in G'\}$$

We will be interested in the relation C_{S_m} where S_m is the canonical set defined below.

Definition A.6 (The canonical set S_m [26])**.** *Let m be a product of distinct primes. The* canonical set S_m *contains all integers $1 \leq i \leq m-1$ that are either 0 or 1 modulo each prime divisor of m. When m is a product of two distinct primes, we have $|S_m| = 3$.*

We will use the following fact:

Fact A.7 ([9])**.** *There exist m, p, β such that there is a share conversion from 3-party CNF sharing over $\mathbb{Z}_m$ to additive sharing over $\mathbb{F}_p^\beta$ with respect to C_{S_m}. In particular, such a conversion exists for the setting $m = 6, p = 2, \beta = 2$.*

B A 2-Server PIR Protocol

In this section, we provide an overview of a concrete 2-server PIR protocol from [CGKS95]. In the following we assume that the database of size n is viewed as a cube (of side length $n^{1/3}$). (Since we let the database to represent an arbitrary function $f : [N] \times [N] \to \{0,1\}$, we will set $n = N^2$.) Let $\mathbf{e}_j$ is the $n^{1/3}$-bit vector which has 0 everywhere except at position j.

For $x, y, z \in \{0,1\}^{n^{1/3}}$ define $G_{\mathrm{CGKS}}(x, y, z) := \bigoplus_{i_1,i_2,i_3 \in [n^{1/3}]} (\langle x, \mathbf{e}_{i_1}\rangle \cdot \langle y, \mathbf{e}_{i_2}\rangle \cdot \langle z, \mathbf{e}_{i_3}\rangle \cdot f(i_1, i_2, i_3))$, where $f(i_1, i_2, i_3)$ is the entry in the database at location (i_1, i_2, i_3) of the cube. That is, $G_{\mathrm{CGKS}}(x, y, z)$ is the exclusive-or of all locations (i_1, i_2, i_3) in the cube satisfying $x_{i_1} = y_{i_2} = z_{i_3} = 1$.

Let i denote the client's real query. Let (u, v, w) denote the position of the i-th entry of the database in the $n^{1/3} \times n^{1/3} \times n^{1/3}$ cube. The PIR protocol proceeds as follows:

- *Query generation algorithm.* The client additively shares $(\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_w) \in \{0,1\}^{3n^{1/3}}$ into $q_1 = (u_1, v_1, w_1)$ and $q_2 = (u_2, v_2, w_2)$, and sends q_1 to $\mathcal{S}_1$ and q_2 to $\mathcal{S}_2$.
- *Answering algorithm.* Each $\mathcal{S}_j$ computes $a_j^0 = G_{\mathrm{CGKS}}(u_j, v_j, w_j)$, and for $\ell \in [n^{1/3}]$, values $a_j^{1,\ell} = G_{\mathrm{CGKS}}(u_j \oplus \mathbf{e}_\ell, v_j, w_j)$, $a_j^{2,\ell} = G_{\mathrm{CGKS}}(u_j, v_j \oplus \mathbf{e}_\ell, w_j)$, $a_j^{3,\ell} = G_{\mathrm{CGKS}}(u_j, v_j, w_j \oplus \mathbf{e}_\ell)$, and sends $a_j = (a_j^0, \{a_j^{c,\ell}\}_{c \in [3], \ell \in [n^{1/3}]})$ to the client.
- *Reconstruction algorithm.* The client reconstructs $z_j = a_j^0 \oplus (\bigoplus_{c \in [3]} a_j^{c,y_c})$ for $j \in \{1, 2\}$, where $(y_1, y_2, y_3) = (u, v, w)$, and outputs $z = z_1 \oplus z_2$.

From the above 2-server PIR protocol, we see that $\gamma(n) = \alpha(n) = 3n^{1/3}$, and that $\beta(n) = 3n^{1/3} + 1$. In other words, $\gamma(n) = \alpha(n) = \beta(n) = \widetilde{O}(N^{2/3})$.

Efficiency of the construction in Figure 1. Let circuit $C(\widetilde{\mathcal{Q}})$ take input x_1, x_2 and form $(\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_w) \in \{0,1\}^{3n^{1/3}}$, where $(\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_w)$ represents the 3-dimensional indicator vector of location in the cube (i.e., (u, v, w)) corresponding to index $x = x_1 \| x_2$, and then using randomness $r = r^{(1)} \oplus r^{(2)}$ additively share $(\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_w)$ into q_1 and q_2, and finally deliver output q_1 to P_1, and q_2 to P_2. It is easy to see that $\tau(C(\widetilde{\mathcal{Q}})) = \tau(C(\mathcal{Q})) = \widetilde{O}(n^{1/3}) = \widetilde{O}(N^{2/3})$.

Let circuit $C(\widetilde{\mathcal{R}})$ select, for each $j \in [2]$ and each $c \in [3]$, value a_j^{c,y_c} from the set $\{a_j^{c,\ell}\}_{\ell \in [n^{1/3}]}$, where $(y_1, y_2, y_3) = (u, v, w)$, and then XOR the selected bits along with a_j^0, and finally deliver this output to both P_1 and P_2. It is easy to see that $\tau(C(\widetilde{\mathcal{R}})) = \tau(C(\mathcal{R})) = \widetilde{O}(n^{1/3}) = \widetilde{O}(N^{2/3})$.

In fact, by a more careful analysis, it is possible to further reduce the size of the circuits from $\widetilde{O}(N^{2/3})$ to $O(N^{2/3})$. We describe this below.

An alternate circuit construction. Let the input of P_1 and P_2 be x_1 and x_2 respectively. Let $(\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_w) \in \{0,1\}^{3n^{1/3}}$ represent the 3-dimensional indicator vector of the location in the cube (i.e., (u, v, w)) corresponding to index $x = x_1 \| x_2$. Note that x_1 completely specifies u, and therefore, party P_1 can locally generate $\mathbf{e}_u$ from index x_1. Similarly, x_2 completely specifies w, and therefore, party P_2 can locally generate $\mathbf{e}_w$ from index x_2. Also, note that x_1 also specifies half the bits of $\mathbf{e}_v$, while x_2 specifies the remaining half of the bits of $\mathbf{e}_v$. Therefore, using x_1, party P_1 can locally generate a string $v_1' \in \{0,1\}^{n^{1/3}}$ in the following way: initialize v_1' to $0^{n^{1/3}}$, then partition v_1' into $n^{1/6}$ consecutive chunks each of size $n^{1/6}$, and set the $\lfloor x/n^{1/6} \rfloor$-th chunk of v_1' to the string $1^{1/6}$. Similarly, using x_2, party P_2 can locally generate a string $v_2' \in \{0,1\}^{n^{1/3}}$ such that for $j \in [n^{1/3}]$, the j-bit of v_2' is set to 1 iff $j \equiv x_2 \bmod n^{1/6}$. Given the above, it is easy to see that $\mathbf{e}_v = v_1' \odot v_2'$, where the notation $a \odot b$ represents the point-wise product (i.e., bit-wise AND) of the two vectors a, b.

Consider an algorithm $\mathcal{Q}'$ which takes as input $(\mathbf{e}_u, v'_1, r_1, r_2)$ from P_1, and $(v'_2, \mathbf{e}_w, r_3)$ from P_2, and computes (1) $u_1 = r_1$, $u_2 = \mathbf{e}_u \oplus r_1$, and (2) $\mathbf{e}_v = v'_1 \odot v'_2$, $v_1 = r_2$, $v_2 = \mathbf{e}_v \oplus r_2$, and (3) $w_1 = \mathbf{e}_w \oplus r_3$, $w_2 = r_3$, and finally outputs $q_1 = (u_1, v_1, w_1)$ to P_1 and $q_2 = (u_2, v_2, w_2)$ to P_2. It is easy to see that the size of the circuit is $n^{1/3} = N^{2/3}$. Observe that q_1 and q_2 are distributed identically as in the 2-server PIR scheme of [20] (in particular, q_1, q_2 form an additive sharing of $(\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_w)$).

Next, consider an algorithm $\mathcal{R}'$ which takes as input $(\mathbf{e}_u, v'_1, (a_1^0, \{a_1^{c,\ell}\}_{c\in[3],\ell\in[n^{1/3}]}), r_1, r_2)$ from P_1, and $(\mathbf{e}_w, v'_2, (a_2^0, \{a_2^{c,\ell}\}_{c\in[3],\ell\in[n^{1/3}]}), r_3)$ from P_2, and performs the following computation. First, it forms $\mathbf{e}_v = v'_1 \odot v'_2$, where $\odot$ is as defined above. Next, for each $c \in [3]$, and for each $j \in [2]$, computes $a_j^c = (a_j^{c,1} \| \cdots \| a_j^{c,n^{1/3}}) \odot \mathbf{e}_{y_c}$, where $(y_1, y_2, y_3) = (u, v, w)$. Finally, it computes $z = a_1^0 \oplus (a_1^1 \oplus a_1^2 \oplus a_1^3) \oplus a_2^0 \oplus (a_2^1 \oplus a_2^2 \oplus a_2^3))$. It is easy to see that algorithm $\mathcal{R}'$ exactly computes the reconstruction algorithm of the 2-server PIR scheme of [20], and that the size of the circuit required to perform the computations above is $6n^{1/3} = O(N^{2/3})$.

From the above, we conclude that the size of the circuits required to emulate the PIR query generation algorithm and the reconstruction algorithm of the 2-server PIR scheme of [20] is $O(N^{2/3})$.

C A 3-Server PIR Protocol

In this section, we provide an overview of a concrete 3-server PIR protocol from [9, 27] which is based on superpolynomial sized matching vector families.

Let $\{u_i\}_{i\in[n]}, \{v_i\}_{i\in[n]}$ with each $u_i, v_i \in \mathbb{Z}_m^h$ for some composite m, denote a S-matching vector family [61, 27, 26] with $S \subseteq \mathbb{Z}_m \backslash \{0\}$. That is, we have

- $\forall i \in [n]$: $\langle u_i, v_i \rangle = 0$
- $\forall i, j \in [n]$, with $j \neq i$: $\langle u_i, v_j \rangle \in S$.

Furthermore, we will use the fact that there is a share conversion procedure from 3-party CNF sharing on $\mathbb{Z}_m$ to 3-party additive sharing on $\mathbb{Z}_p^\beta$ for the relation defined by $C_S = \{(s, 0) : s \in S\} \cup \{(0, s') : s' \in \mathbb{Z}_p^\beta \backslash \{0\}\} \cup \{(s, s') : s \notin S \cup \{0\}, s' \in \mathbb{Z}_p^\beta\}$ (cf. Fact A.7).

Let n denote the size of the database. (Since we let the database to represent an arbitrary function $f : [N] \times [N] \to \{0, 1\}$, we will set $n = N^2$.) Let i denote $\mathcal{U}$'s real query. The PIR protocol proceeds as follows:

- $\mathcal{U}$ shares $u' = u_i \in \mathbb{Z}_m^h$ into $u'_1, u'_2, u'_3 \in \mathbb{Z}_m^h$, and sends CNF share (u'_{j-1}, u'_{j+1}) to $\mathcal{S}_j$.
- Each $\mathcal{S}_j$ computes $a_{j,j-1,\ell} = \langle u'_{j-1}, v_\ell \rangle$ and $a_{j,j+1,\ell} = \langle u'_{j+1}, v_\ell \rangle$ for each $\ell \in [n]$. Then it applies the share conversion procedure to each CNF share $(a_{j,j-1,\ell}, a_{j,j+1,\ell}) \in \mathbb{Z}_m^2$ to obtain $a'_{j,\ell} \in \mathbb{Z}_p^\beta$. Finally, it sends $y_j = \sum_{\ell\in[n]} x_\ell \cdot a'_{j,\ell}$ to $\mathcal{U}$. Here, $x_\ell \in \{0, 1\}$ denotes the ℓ-th element in the database, and is interpreted as an element in $\mathbb{Z}_p$.
- $\mathcal{U}$ reconstructs $z = y_1 + y_2 + y_3$, and outputs 0 if $z = 0 \in \mathbb{Z}_p^\beta$, else outputs 1.

From the above protocol, we see that when m, p, β are small constants (e.g., $m = 6, p = \beta = 2$; see Fact A.7) , we have that $\gamma(n) = \alpha(n) = O(h)$, and that $\beta(n) = O(1)$. Since $\beta(n) = O(1)$, we have $\tau(C(\widetilde{\mathcal{R}})) = O(1)$. In the following, we will show that $h = 2^{\widetilde{O}(\sqrt{\log n})}$, thereby bounding $\tau(C(\widetilde{\mathcal{Q}}))$ by $2^{\widetilde{O}(\sqrt{\log n})} = 2^{\widetilde{O}(\sqrt{\log N})}$.

C.1 Parameters of Matching Vector Families

We will be interested in an explicit construction of a S-matching vector family in $\mathbb{Z}_6^h$ with the goal of demonstrating that there exists a circuit of size $\widetilde{O}(h)$ that takes any $i \in [n]$ as input and outputs the i-th matching vector $u_i \in \mathbb{Z}_6^h$.

We will use the following facts:

- There exists a share conversion procedure from 3-party CNF on $\mathbb{Z}_6$ to 3-party additive share on $\mathbb{Z}_2^2$ with respect to C_S for $S = \{1, 3, 4\}$ [9].
- There exists a S-matching vector family in $\mathbb{Z}_6^h$ of size n such that $h = 2^{\widetilde{O}(\sqrt{\log n})}$ [34, 27, 26].

In fact, we will need to show an explicit S-matching vector family in $\mathbb{Z}_6^h$. We use the following result from [26].

Lemma C.1 (Corollary 41, [26]). *Let $m = \Pi_{i=1}^{t} p_i$ be a product of distinct primes. Let w be a positive integer. Suppose integers $\{e_i\}_{i\in[t]}$ are such that for all i, we have $p_i^{e_i} > w^{1/t}$. Let $d = \max_i p_i^{e_i}$, and $k \geq w$ be arbitrary. Let S be the canonical set modulo m. There is an explicit multilinear polynomial $f(z_1, \ldots, z_k) \in \mathbb{Z}_m[z_1, \ldots, z_k]$, $\deg(f) \leq \max_{i\in[t]} (p_i^{e_i} - 1)$ such that for all $\mathbf{x} \in \{0,1\}^k$, we have*

$$f(x) = \begin{cases} 0 \bmod m, & \text{if } \sum_{\ell=1}^{k} \mathbf{x}(\ell) = w \\ s \bmod m, \text{ for } s \in S, & \text{if } \sum_{\ell=1}^{k} \mathbf{x}(\ell) < w \end{cases}$$

where coordinates of $\mathbf{x}$ are summed as integers.

Following the presentation in [26], we now describe how to derive a S-matching vector family using the explicit multilinear polynomial f given by Lemma C.1.

For every $T \subseteq [k]$ of size w, define the polynomial f_T which is the polynomial f from Lemma C.1 with z_j set to 0 for $j \notin T$. Define $\mathbf{x}_T \in \{0,1\}^k$ to be the indicator of the set T. Viewing vectors $\mathbf{x} \in \{0,1\}^k$ as indicator vectors $\mathbf{x}_L$ for sets $L \subseteq [k]$ it is easy to check that for all $T, L \in [k]$, $f_T(\mathbf{x}_L) = f(\mathbf{x}_{L\cap T})$. Combining this with Lemma C.1 gives

- For all $T \subseteq [k]$, where $|T| = w$, $f_T(\mathbf{x}_T) = f(\mathbf{x}_T) \equiv 0 \bmod m$.
- For all $T \neq L \subseteq [k]$, where $|T| = |L| = w$, $f_T(\mathbf{x}_L) = f(\mathbf{x}_{L\cap T}) \in S \bmod m$.

For $i \in [n]$ with $n = \binom{k}{w}$, let T_i denote the i-th subset of $[k]$ with Hamming weight w. Let u_i be the vector of coefficients of f_{T_i}. Let v_i be the evaluation of monomials of f at the point $\mathbf{x}_{T_i}$. Note $|u_i| = |v_i| = \binom{k}{\leq d}$, and that for $i, j \in [n]$, we have $\langle u_i, v_j \rangle = f_{T_i}(\mathbf{x}_{T_j})$. It is easy to see that $\{u_i\}_{i\in[n]}$ and $\{v_i\}_{i\in[n]}$ form $\binom{k}{w}$-sized family of S-matching vectors in $\mathbb{Z}_m^h$ where $h = \binom{k}{\leq d}$. Now we set $t = 2, p_1 = 2, p_2 = 3$, and let w grow to infinity. Choose e_1, e_2 to be the smallest integers satisfying $2^{e_1} > \sqrt{w}$ and $3^{e_2} > \sqrt{w}$. Clearly, $d = \max(2^{e_1}, 3^{e_2}) \leq 3\sqrt{w}$. Setting $k = w^2$, we see that $n = \binom{w^2}{w}$, and $h = \binom{w^2}{3\sqrt{w}}$. Thus, $\log n = \Theta(w \log w)$, and $\log h = \Theta(\sqrt{w} \log w)$. We conclude that $h = 2^{\widetilde{O}(\sqrt{\log n})}$.

Efficiency of the construction in Figure 2. We saw earlier that $\tau(C(\widetilde{\mathcal{R}})) = O(1)$. Now we will show that $\tau(C(\widetilde{\mathcal{Q}})) = 2^{\widetilde{O}(\sqrt{\log N})}$. We first bound the size of the circuit that takes $i = (x_1, x_2)$ as input and outputs u_i, i.e., the i-th matching vector. From above, we have that u_i represents the vector of coefficients of f_{T_i}, where T_i denotes the i-th subset of $[k]$ with Hamming weight w. We make the following observations:

- Given i, the indicator vector representing T_i, i.e., $\mathbf{x}_{T_i}$, can be computed using a circuit of size $O(\mathrm{poly}(w)) = O(\mathrm{polylog}(h))$.
- Given coefficients of f (i.e., the k-variate multilinear polynomial of total degree d specified by Lemma C.1) and $\mathbf{x}_{T_i}$, the vector of coefficients of f_{T_i} can be computed using a circuit of size $O(c(f) \cdot d \cdot k)$, where $c(f) = \binom{k}{\leq d}$ represents the number of coefficients of f. Rewriting in terms of w, we see that the size of this circuit is $O(\binom{w^2}{3\sqrt{w}} \cdot 3\sqrt{w} \cdot w^2) = O(h\mathrm{polylog}(h))$.

Therefore, we conclude that the size of the circuit that takes $i \in [n]$ as input and outputs the i-th matching vector u_i is of size $O(h\mathrm{polylog}(h)) = 2^{\widetilde{O}(\sqrt{\log n})}$. In other words, $\tau(C(\widetilde{\mathcal{Q}})) = \tau(C(\mathcal{Q})) = 2^{\widetilde{O}(\sqrt{\log N})}$.

D 3-Round Secure Computation in the Preprocessing Model

The protocols of Figures 1 and 2 use semi-honest secure GMW as a subroutine and thus their round complexity equals the sum of the depths of circuits $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$. If, instead of using GMW protocol as a subroutine, one uses information-theoretic variants of Yao's garbled circuit construction [55, 49, 21, 40], then it is possible to reduce the round complexity to 4 while preserving, up to logarithmic factors, the communication complexity and the correlated randomness complexity of the protocols of Figures 1 and 2. (In more detail, circuits $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$ can each be evaluated in 2 rounds using information theoretic garbled circuits in the OT hybrid/preprocessing model. In the OT hybrid model, we assume that the call to the OT oracle takes one round, and the response from the oracle takes another round. In the OT preprocessing model, an ideal OT oracle can be emulated using precomputed OTs by a 2-round protocol [3].)

We remark that there exists 2-round protocol for secure computation in the preprocessing model [42], but the correlated randomness complexity of their construction is $O(N^2)$. It is also possible to naturally formulate

the secure evaluation of $f : [N] \times [N] \to \{0,1\}$ as a 1-out-of-N OT protocol between the two parties, which in turn can be realized by $O(N)$ instances of 1-out-of-2 OT [16]. Such a transformation would yield a 2-round protocol whose correlated randomness complexity is $O(N)$.

In this section, we show a 3-round protocol whose communication and correlated randomness complexity is $O(N^{1/2})$. Our construction is inspired by our PSM protocol presented in Section 6.3, which in turn is inspired by the 4-server PIR scheme of [20]. Formally, we prove the following theorem.

Theorem D.1. *For any 2-party functionality $f : [N] \times [N] \to \{0,1\}$, there is a 3-round protocol π which realizes f in the preprocessing model, that is perfectly secure against semi-honest parties, and has total communication complexity and correlated randomness complexity $O(N^{1/2})$.*

Proof. Recall that in the 4-server PIR scheme of [20], the N^2-bit database representing the truth table of $f : [N]\times[N] \to \{0,1\}$ is associated with the 4-dimensional cube $[N^{1/2}]^4$, where each position $x = (x_1, x_2) \in [N] \times [N]$ in the truth-table is associated with a 4-tuple $(\hat{x}_1, \ldots, \hat{x}_4) \in [N^{1/2}]^4$, in a natural manner. The 4-server PIR scheme proceeds by letting the client additively share $(\mathbf{e}_{\hat{x}_1}, \ldots, \mathbf{e}_{\hat{x}_4})$ into two $O(N^{1/2})$-sized bit vectors (u_0, v_0, w_0, z_0) and (u_1, v_1, w_1, z_1). For $j_1, j_2, j_3, j_4 \in \{0,1\}$, the client prepares queries $q_{j_1j_2j_3j_4} = (u_{j_1}, v_{j_2}, w_{j_3}, z_{j_4})$ to send to a *virtual* server $\mathcal{S}_{j_1j_2j_3j_4}$. These 16 virtual servers are in turn emulated by the 4 actual servers.

We now proceed to describe our 2-party secure computation protocol in the preprocessing model that has communication complexity $O(N^{1/2})$ and round complexity 3.

- *Correlated randomness.* Let query $q_{1111} = (u_1, v_1, w_1, z_1)$. Then (u_1, v_1) is given to P_1 and (w_1, z_1) is given to P_2. The 1-bit answer to that query $a_{1111} = \mathcal{S}_{1111}(f, q_{1111})$ is additively shared among the two parties. Further, parties P_1 and P_2 are provided with OT correlations of size $O(N^{1/2})$.
- *Round 1.* Recall that $(\hat{x}_1, \ldots, \hat{x}_4) \in [N^{1/2}]^4$ denotes the position in the 4-dimensional cube that is associated with $x = (x_1, x_2) \in [N] \times [N]$. Recall that input x_1 completely defines $\hat{x}_1, \hat{x}_2$, and in turn $\mathbf{e}_{\hat{x}_1}, \mathbf{e}_{\hat{x}_2}$. Similarly, input x_2 completely defines $\hat{x}_3, \hat{x}_4$, and in turn $\mathbf{e}_{\hat{x}_3}, \mathbf{e}_{\hat{x}_4}$. Using (u_1, v_1) and x_1 (which defines $\mathbf{e}_{\hat{x}_1}, \mathbf{e}_{\hat{x}_2}$), P_1 can therefore generate (u_0, v_0) which it then sends to P_2. Similarly, using (w_1, z_1) and x_2 (which defines $\mathbf{e}_{\hat{x}_3}, \mathbf{e}_{\hat{x}_4}$), P_2 can generate (w_0, z_0) which it then sends to P_1. Thus, both parties now possess $q_{0000} = (u_0, v_0, w_0, z_0)$. In addition, note that P_1 possesses (u_1, v_1), while P_2 possesses (w_1, z_1).
- *Local Simulation of PIR Servers.* Using q_{0000} and values (u_1, v_1), for each $j_1, j_2 \in \{0,1\}$, party P_1 can locally simulate servers $\mathcal{S}_{j_1j_200}$, and also generate locally a list of size $O(N^{1/2})$ for servers $\mathcal{S}_{j_1j_201}$ (i.e., by iterating over $N^{1/2}$ possible values of $\mathbf{e}_{\hat{x}_4}$) and $\mathcal{S}_{j_1j_210}$ (i.e., by iterating over $N^{1/2}$ possible values of $\mathbf{e}_{\hat{x}_3}$).

  Similarly, using q_{0000} and values (w_1, z_1), for each $j_3, j_4 \in \{0,1\}$, party P_2 can locally simulate $\mathcal{S}_{00j_3j_4}$ and also generate locally a list of size $O(N^{1/2})$ for servers $\mathcal{S}_{10j_3j_4}$ (i.e., by iterating over $N^{1/2}$ possible values of $\mathbf{e}_{\hat{x}_1}$) and $\mathcal{S}_{01j_3j_4}$ (i.e., by iterating over $N^{1/2}$ possible values of $\mathbf{e}_{\hat{x}_2}$).

  Note that at this point, party P_1 holds, for each $j_1, j_2 \in \{0,1\}$, values $a_{j_1j_200}, \{a^{\ell}_{j_1j_201}\}_{\ell\in[N^{1/2}]}$, $\{a^{\ell}_{j_1j_210}\}_{\ell\in[N^{1/2}]}$, while party P_2 holds, for each $j_3, j_4 \in \{0,1\}$, values $a_{00j_3j_4}, \{a^{\ell}_{01j_3j_4}\}_{\ell\in[N^{1/2}]}$, $\{a^{\ell}_{10j_3j_4}\}_{\ell\in[N^{1/2}]}$.
- *Rounds 2-3.* In this step, parties engage, via a information theoretic Yao garbled circuit protocol [49, 21, 55, 40], to securely evaluate the following steps: (1) reconstruct a_{1111} using the shares provided by P_1 and P_2, and (2) reconstruct $(\mathbf{e}_{\hat{x}_1}, \mathbf{e}_{\hat{x}_2}, \mathbf{e}_{\hat{x}_3}, \mathbf{e}_{\hat{x}_4}) = (u_0 \oplus u_1, v_0 \oplus v_1, w_0 \oplus w_1, z_0 \oplus z_1)$, and (3) for each $j_1, j_2 \in \{0,1\}$, reconstruct $a_{j_1j_201} = (a^1_{j_1j_201} \| \cdots \| a^{N^{1/2}}_{j_1j_201}) \odot \mathbf{e}_{\hat{x}_4}$, and $a_{j_1j_210} = (a^1_{j_1j_210} \| \cdots \| a^{N^{1/2}}_{j_1j_210}) \odot \mathbf{e}_{\hat{x}_3}$, and (4) for each $j_3, j_4 \in \{0,1\}$, reconstruct $a_{01j_3j_4} = (a^1_{01j_3j_4} \| \cdots \| a^{N^{1/2}}_{01j_3j_4}) \odot \mathbf{e}_{\hat{x}_2}$, and $a_{10j_3j_4} = (a^1_{10j_3j_4} \| \cdots \| a^{N^{1/2}}_{10j_3j_4}) \odot \mathbf{e}_{\hat{x}_1}$, and (5) compute $z = \bigoplus_{j_1,j_2,j_3,j_4\in\{0,1\}} a_{j_1j_2j_3j_4}$, and deliver z to both P_1 and P_2.

  Note that this step requires parallel invocations of 2-message OT (which is computed using precomputed OT correlations [3] provided to both parties), i.e., 2 rounds of interaction. It is easy to see that the circuit representing the computation above is of depth-1 and consists of $O(N^{1/2})$ AND gates. Thus, the information-theoretic Yao garbled circuit construction requires communication (including oblivious transfer) of $O(N^{1/2})$ bits.

□

A natural open question is whether it is possible to construct 2-round protocol for 2-party secure computation in the preprocessing model with communication complexity $o(N)$.

E The FKN Construction

In this section, we show how the framework of decomposable PIR captures the PSM scheme of Feige et al. [30]. We begin with a review of the FKN construction.

The FKN construction:

Preliminaries. Let P_1, P_2 denote the clients and let R denote the referee. We assume that parties P_1, P_2 have inputs $x_1, x_2 \in [N]$ respectively. At the end of the protocol, we want the referee to learn $z = f(x_1, x_2)$ for an arbitrary $f : [N] \times [N] \to \{0, 1\}$.

Shared randomness: Both parties share uniform randomness:

- $r = (r_1, \ldots, r_N) \in \{0, 1\}^N$.
- $s \in [N]$.

Protocol:

1. Using its input x_1 and shared randomness $r = (r_1, \ldots, r_N)$, party P_1 computes $c_j = f(x_1, j) \oplus r_j$ for all $j \in [N]$. Then using shared randomness s, party P_1 computes $d_j = c_{j+s}$ for all $j \in [N]$. (Here $j + s$ is computed modulo N.) Finally, P_1 sends $\{d_j\}_{j\in[N]}$ to R.

   Similarly, using its input x_2 and shared randomness r, party P_2 computes $r' = r_{x_2}$, and using shared randomness s, computes $s' = x_2 - s$ (where the subtraction is done modulo N). Party P_2 sends (r', s') to R.
2. The referee R computes $z = r' \oplus d_{s'}$, and outputs z.

We now present a decomposable PIR scheme that corresponds to the FKN construction described above.

- *Decomposable query generation algorithm.* Let $r = (r_1, \ldots, r_N)$ be a uniformly random string of length N. Let s denote a random element in $[N]$. We let $u = (r, s)$ denote the randomness required for PIR query generation algorithm. The query generation algorithm is described as follows.
  1. Algorithm $\mathcal{Q}_1(x_1, u)$: Output $q_1 = (x_1, u)$.
  2. Algorithm $\mathcal{Q}_2(x_2, u)$: Output $q_2 = 0$.
  3. Algorithm $\mathcal{Q}_3(x_1, x_2, u)$:
     - Algorithm $\mathcal{Q}_3^1(x_1, u)$: Output $q_3^1 = 0$.
     - Algorithm $\mathcal{Q}_3^2(x_2, u)$: Output $q_3^2 = 0$.
- *Answering algorithm.* The answering algorithm is described as follows:
  1. Algorithm $\mathcal{A}(1, q_1 = (x_1, u), f)$: Parse u as (r, s) where $r \in \{0, 1\}^N$ and $s \in [N]$. Then, perform the following:
     - For all $j \in [N]$, compute $c_j = f(x_1, j) \oplus r_j$.
     - For all $j \in [N]$, compute $d_j = c_{j+s}$.
     - Output $a_1 = \{d_j\}_{j\in[N]}$.
  2. Algorithm $\mathcal{A}(2, q_2, f)$: Output $a_2 = 0$.
  3. Algorithm $\mathcal{A}(3, q_3 = (q_3^1, q_3^2), f)$: Output $a_3 = 0$.
- *Decomposable reconstruction algorithm.*
  - Algorithm $\mathcal{R}'((x_1, x_2), u, a_1, a_2)$:
    - Parse a_1 as $\{d_j\}_{j\in[N]}$.
    - Parse u as (r, s) where $r \in \{0, 1\}^N$ and $s \in [N]$.
    - Compute $s' = x_2 - s$.
    - Output $z' = d_{s'} \oplus r_{x_2}$.
  - Algorithm $\mathcal{R}''(a_3, z')$: Output $z = a_3 \oplus z' = z'$.

F Secure Sampling

We concern ourselves with pairs of correlated *finite* random variables (X, Y) with joint distribution p_{XY}. $\mathcal{X}$ and $\mathcal{Y}$ shall stand for the (finite) alphabets of X and Y respectively. The definitions we present below are inspired by the definitions in [53].

Definition F.1 (Ideal Sampler Functionality)**.** *For a pair of correlated finite random variables (X, Y) with joint distribution p_{XY}, and where $\mathcal{X}$, $\mathcal{Y}$ represents the (finite) alphabets of X and Y, a* 2-party ideal sampler functionality for a pair of correlated random variables (X, Y)*, denoted by $\mathcal{G}_{\text{samp}}^{X,Y}$ interacts with two parties P_1, P_2 in the following way: (1) Upon receiving a* request *from both parties, it chooses $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with probability $p_{XY}(x, y)$, and (2) sends x to P_1 and y to P_2.*

Definition F.2 (Secure Sampling)**.** *We say that a distribution D, consisting of a pair of correlated random variables (U, V) can be* securely sampled *with error at most ϵ, using a pair of correlated random variables (X, Y) as setup if there exists a 2-party protocol π that, except with probability ϵ (over the random coins of π), securely realizes $\mathcal{G}_{\text{samp}}^{U,V}$ (cf. Definition F.1) while allowing the two parties to request samples from $\mathcal{G}_{\text{samp}}^{X,Y}$.*

Definition F.3. *A pair of correlated random variables (X, Y) is called a* random OT correlation *if $(X, Y) = ((X_0, X_1), (b, X_b))$ where X_0, X_1, b are uniformly random variables in $\{0, 1\}$.*

We prove the following theorem.

Theorem F.4. *Let D be a distribution over $[N] \times [N]$ such that for each $(x, y) \in [N] \times [N]$, the probability of sampling (x, y) from D can be expressed as a rational number whose denominator equals d. Then, for every $\epsilon > 0$, it is possible to securely sample from D with error at most ϵ using at most $O(N^{2/3}\text{poly}(\log N, \log d, \log(1/\epsilon)))$ random OT correlations (alternatively, calls to an OT oracle).*

Proof. Our strategy is to obtain a sample from D in a bit-by-bit fashion. Towards this, we define distributions $\{D_{|i}\}_{i \in \{0,1,\ldots,\log N\}}$ derived from D in the following way. For each $i \in \{0, 1, \ldots, \log N\}$, let $D_{|i}$ denote the distribution which is obtained by sampling pairs of strings from distribution D, and then truncating the sample to contain only the first i bits of each string in the sampled pair. Then, for each possible prefix $x_{|i}, y_{|i} \in \{0, 1\}^i$, define

$$g^D_{x_{|i},y_{|i}}(b_1, b_2) = \left\lfloor \left(\frac{\Pr\left[(x_{|i} \| b_1, y_{|i} \| b_2) \leftarrow D_{|i+1}\right]}{\Pr\left[(x_{|i}, y_{|i}) \leftarrow D_{|i}\right]} \right) \cdot d2^k \right\rfloor,$$

where $k = \log\left(\frac{4 \log N}{\epsilon}\right)$. Next, consider a function $h^D_i : \{0, 1\}^i \times [d2^k] \times \{0, 1\}^i \times [d2^k] \to \{0, 1\}^2$ that takes $x_{|i}, y_{|i} \in \{0, 1\}^i$, $r_1, r_2 \in [d2^k]$, outputs $(x_{|i} \| b_1, y_{|i} \| b_2)$ such that

$$\left(\sum_{c_1, c_2 \,:\, 2b_1 + b_2 > 2c_1 + c_2} g^D_{x_{|i},y_{|i}}(c_1, c_2) \right) \leq (r_1 + r_2 \bmod d2^k) \leq \left(\sum_{c_1, c_2 \,:\, 2b_1 + b_2 \geq 2c_1 + c_2} g^D_{x_{|i},y_{|i}}(c_1, c_2) \right),$$

and if no such (b_1, b_2) exists, then outputs $(x_{|i} \| 1, y_{|i} \| 1)$. Loosely speaking, we will show that starting from a sample $(x_{|i}, y_{|i}) \leftarrow D_{|i}$, the function h^D_i lets us sample from a distribution that is statistically close to $D_{|i+1}$.

More formally, for $i \in \{0, \ldots, \log N - 1\}$, let $D'_{|i+1}$ denote the distribution which is obtained by (1) first sampling $(x_{|i}, y_{|i})$ from distribution $D_{|i}$, and then (2) choosing a random $r^{(i)}_1, r^{(i)}_2 \in [d2^k]$, and (3) outputting $h^D_i((x_{|i}, r^{(i)}_1), (y_{|i}, r^{(i)}_2))$ as a sample of $D_{|i+1}$. For $(x_{|i+1}, y_{|i+1}) \in \{0, 1\}^{i+1} \times \{0, 1\}^{i+1}$, let $P_D(x_{|i+1}, y_{|i+1}) = \Pr[(x_{|i+1}, y_{|i+1}) \leftarrow D_{|i+1}]$. Similarly, define $P_{D'}(x_{|i+1}, y_{|i+1})$ as $\Pr[(x_{|i+1}, y_{|i+1}) \leftarrow D'_{|i+1}]$, where the probability is over the choice of randomness $r^{(i)}_1, r^{(i)}_2$ used in function h^D_i.

We now claim that the statistical distance between the distributions $D_{|i+1}$ and $D'_{|i+1}$ is at most $\epsilon / \log N$. First note that by the assumption in the theorem, for all $(x, y) \in [N] \times [N]$, the value $\Pr[(x, y) \leftarrow D]$ is a rational number whose denominator is bounded by d. Given this, it is easy to see that for all $(x_{|i}, y_{|i}) \in \{0, 1\}^i \times \{0, 1\}^i$, the value $\Pr[(x_{|i}, y_{|i}) \leftarrow D_{|i}]$ is also a rational number whose denominator is bounded by d. This in turn implies for all $b_1, b_2 \in \{0, 1\}$, the value $\frac{\Pr[(x_{|i} \| b_1, y_{|i} \| b_2) \leftarrow D_{|i+1}]}{\Pr[(x_{|i}, y_{|i}) \leftarrow D_{|i}]}$ is also a rational number whose denominator is bounded by d. Thus, for each possible prefix $x_{|i}, y_{|i} \in \{0, 1\}^i$, and for each $b_1, b_2 \in \{0, 1\}$, the value $g^D_{x_{|i},y_{|i}}(b_1, b_2)$ approximates the exact probability of sampling $(x_{|i} \| b_1, y_{|i} \| b_2)$ from $D_{|i+1}$ conditioned

on the sample (x, y) from D having $x_{|i}$ (resp. $y_{|i}$) as the length-i prefix of x (resp. y), up to an additive error $2^{-k} = \epsilon/4 \log N$. Therefore, the statistical distance between distributions $D_{|i+1}$ and $D'_{|i+1}$ is at most $\epsilon/\log N$.

The discussion above naturally motivates the following construction of a secure sampling protocol.

Protocol π^D_{samp}.

1. P_1 initializes x_0 to be the empty boolean string, and P_2 initializes y_0 to be the empty boolean string.
2. For $i = 0, 1, 2, \ldots, \log N - 1$, parties P_1 and P_2 perform the following:
   (a) P_1 chooses random $r_1^{(i)} \in [d2^k]$. Similarly, P_2 chooses random $r_2^{(i)} \in [d2^k]$.
   (b) P_1 and P_2 securely evaluate $h_i^D((x_{|i}, r_1^{(i)}), (y_{|i}, r_2^{(i)}))$ to obtain their respective outputs $x_{|i+1}, y_{|i+1} \in \{0,1\}^{i+1}$.
3. P_1 outputs $x' = x_{|\log N}$. P_2 outputs $y' = y_{|\log N}$.

For each $i \in \{0, 1, \ldots, \log N - 1\}$, let $D''_{|i+1}$ denote the distribution of $(x_{|i+1}, y_{|i+1})$ computed by the parties in Step 2b of the protocol π^D_{samp}. By a standard application of triangle inequality, we can conclude that statistical distance between $D''_{|i+1}$ and $D_{|i+1}$ is at most $(i+1)\epsilon/\log N$. Therefore, the statistical distance between $D''_{|\log N}$ and $D_{|\log N} = D$ is at most ϵ.

Efficiency. Secure evaluation of h_i^D in Step 2b requires securely evaluating 4 invocations of $g^D_{x_{|i}, y_{|i}}$ (and then choosing the right value of the bits to be appended based on random values $r_1^{(i)}$ an $r_2^{(i)}$). To maximize efficiency of our sampling protocol, we instantiate the secure evaluation of invocations of $g_{x_{|i}, y_{|i}}$ using the protocol in Figure 1. Observe that using Theorem 3.1, that Step 2b in the i-th iteration can be performed using $\widetilde{O}(i^{2/3} \log(d2^k))$ OT correlations (alternatively calls to an OT oracle). Since i varies from 1 through $\log N$, we have that the total number of OT correlations (alternatively, calls to the OT oracle) required by the sampling protocol is bounded by $O(N^{2/3}\text{poly}(\log N, \log d, \log(1/\epsilon)))$. □

G Computing Functions of a Shared Secret

The work of Beimel et al. [6] introduced threshold (t-out-of-n) secret-sharing schemes *for families of functions* $\mathcal{F}$. Such schemes allow any set of at least t parties to compute privately the value $f(s)$ of a (previously distributed) secret s, for any $f \in \mathcal{F}$. Smaller sets of players get no more information about the secret than what follows from the value $f(s)$. The goal is to make the shares as short as possible. A key feature of such schemes is that after the computation of $f(s)$ by some set consisting of at least t parties, no information about s is leaked to any coalition of less than t parties other than what was implied by the value of $f(s)$.

Model and definitions. Let $U = \{P_1, \ldots, P_n\}$ denote the set of all parties. In addition to the parties, there is a dealer who has a secret input s. A distribution scheme is a probabilistic mapping, which the dealer applies to the secret to generate n pieces of information $s_1, \ldots, s_n$ which are referred to as the shares. The dealer gives the share s_i to party P_i. In the scenario we consider, the dealer is active only during the initialization of the system. After this stage the parties can communicate. We stress that the particular function $f \in \mathcal{F}$ is chosen *after* the initialization of the system by the dealer.

Let $B \subseteq U$ of size at least t as the set of parties that are required to compute the value of $f(s)$. For any coalition $C \subset U$ of size less than t, we denote by $M_C(\langle s_i \rangle_B, \langle r_i \rangle_B)$ the messages that an eavesdropping coalition C can obtain during the communication between the parties in B. Here $\langle s_i \rangle_B$ is the vector of shares of B and $\langle r_i \rangle_B$ the vector of random inputs of B.

Definition G.1 (View of a coalition [6])**.** *Let $C \subseteq U$ be a coalition. The view of C, denoted $VIEW_C$, after the execution of a protocol consists of the information that C gains. In a distribution scheme, $VIEW_C = \langle s_i \rangle_C$, the shares of C. In the evaluation of $f(s)$ by a set of parties $B \subseteq U$ the view of C consists of the inputs $\langle s_i \rangle_C$, the local random inputs $\langle r_i \rangle_{C \cap B}$ (only the parties in $C \cap B$ are involved in the computation), and the messages that C obtains from the communication channel during the evaluation by B, i.e., $M_{B \cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B)$. That is,*

$$VIEW_C = \langle s_i \rangle_C \| \langle r_i \rangle_{i \in B \cap C} \| M_{B \cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B).$$

Definition G.2 (Secret-sharing schemes for a family of functions [6])**.** *A t-out-of-n secret-sharing scheme for a family of functions $\mathcal{F}$ is a distribution scheme (Π, μ) which satisfies the following two conditions.*

Evaluation: For any set $B \subseteq U$ of size at least t and any function $f \in \mathcal{F}$ the parties in B can evaluate $f(s)$. That is, there is a protocol $F_{B,f}$ which, given the shares of B as inputs, will always output the correct value of $f(s)$.

Security: Let X be a random variable on the set of secrets S and $C \subset U$ be any coalition of size less than t.

Prior to evaluation (after the distribution of shares). *For any two secrets $s, s' \notin S$ and any shares $\langle s_i \rangle_C$ (from $\Pi(s, r)$):*

$$\Pr[VIEW_C = \langle s_i \rangle_C \mid X = s] = \Pr[VIEW_C = \langle s_i \rangle_C \mid X = s].$$

After evaluation. *For any $f \in \mathcal{F}$, any $B \subseteq U$ of size at least t, any secrets $s, s \in S$ with $f(s) = f(s)$, any shares $\langle s_i \rangle_C$, any inputs $\langle r_i \rangle_{B \cap C}$, and any messages $M_{B \cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B)$ from the computation of $f(s)$ by B:*

$$\begin{aligned}&\Pr[VIEW_C = \langle s_i \rangle_C \| \langle r_i \rangle_{B \cap C} \| M_{B \cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) | X = s] \\ &= \Pr[VIEW_C = \langle s_i \rangle_C \| \langle r_i \rangle_{B \cap C} \| M_{B \cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) | X = s].\end{aligned}$$

We say that C gains no information that is not implied by the function f *(or simply: gains no additional information) if in the computation of $f(s)$ we have security after evaluation.*

Among other things, Beimel et al. [6] prove that for the class of all functions $f : \{0,1\}^\ell \to \{0,1\}$, denoted $\mathcal{ALL}_\ell$, there exists a t-out-of-n (interactive) secret-sharing scheme *over private communication channels* for $\mathcal{ALL}_\ell$ with shares of length $\max(2^\ell - 1, \log n + 1)$. They pose as an explicit open question as to whether there exists a better scheme for the family $\mathcal{ALL}_\ell$.

Our contribution. Loosely speaking, at a very high level, we show that the *number of OTs* required to evaluate any function $f \in \mathcal{ALL}_\ell$ among t parties such that the computation remains private from any coalition of less than t parties is (up to constant factors) an *upper bound* on the size of shares of secret-sharing schemes for $\mathcal{ALL}_\ell$. Then, by a straightforward generalization of our results in Section 4, we obtain improved upper bounds on the share length of t-out-of-n interactive secret-sharing scheme over private communication channels for $\mathcal{ALL}_\ell$. The generalization involves use of $(t-1)$-private t-server PIR schemes (instead of 1-private PIR schemes as in Definition 2.2) which, loosely speaking preserves privacy of client input even against coalitions of $t-1$ (or less) out of the t servers. As before, we take a non-standard view of PIR protocols in which the database held by the servers corresponds to the truth table of some function $f : \{0,1\}^\ell \to \{0,1\}$.

We obtain, in particular for the simplest case of $n = 2$, $t = 2$, a secret-sharing scheme for $\mathcal{ALL}_\ell$ whose share length is at most $\widetilde{O}(2^{\ell/3})$ for $\ell = \omega(\mathrm{polylog}(n))$. Formally, and more generally, we prove the following theorem.

Theorem G.3. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a $(t-1)$-private t-server PIR scheme for the function class $\mathcal{ALL}_\ell$. Let $\tau(\cdot)$ be as in Definition 2.1. Further, let $\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}}$ denote the reconstruction algorithm of the $(t-1)$-private t-reconstructible n-party Shamir secret-sharing that accepts shares from t parties and reconstructs a secret $s \in \{0,1\}^\ell$. Then, there exists a t-out-of-n interactive secret-sharing scheme over private communication channels for $\mathcal{ALL}_\ell$ with shares of length $O(n^2(\tau(\mathcal{Q}) + \tau(\mathcal{R}) + \tau(\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}})))$.*

Proof. The dealer performs the following:

1. The dealer shares its input secret $s \in \{0,1\}^\ell$ using the $(t-1)$-private t-reconstructible n-party Shamir secret-sharing scheme.
2. The dealer distributes random OT correlations of size $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}) + \tau(\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}}))$ for each pair of parties, where $\mathcal{Q}$ and $\mathcal{R}$ are, respectively, the query generation and reconstruction algorithm of the given $(t-1)$-private t-server PIR scheme $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$.

This completes the description of the initialization of the system.

After this, suppose any set $B \subseteq U$ of size at least t agree on a function $f \in \mathcal{ALL}_\ell$. In the following, we assume that the size of B is t. This is without loss of generality since for any set of size greater than t, we may just choose (the lexicographically first) t parties in that set to reconstruct $f(s)$ and then distribute this value over private point-to-point channels to all other parties.

To compute $f(s)$, parties in B perform the following:

1. Each $P_i \in B$ with input share s_i participates to securely evaluate, using semi-honest secure GMW protocol, a circuit that
   - reconstructs secret $s \in \{0,1\}^\ell$ using $\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}}$ on input $\langle s_i \rangle_B$, and

- accepts random $r_i \in \{0,1\}^{\gamma(N)}$ chosen by each $P_i \in B$, where $\gamma(N)$ denotes the size of randomness required by the query generation algorithm $\mathcal{Q}$, and computes $r = \bigoplus_{P_i \in B} r_i$, and
- computes $(q_1, \ldots, q_t) \leftarrow \mathcal{Q}(s, r)$, and
- delivers output q_j to party $P_j \in B$.

2. In the next phase, each party $P_i \in B$ computes $a_i \leftarrow \mathcal{A}(i, q_i, f)$.
3. Each $P_i \in B$ with input share s_i participates to securely evaluate, using semi-honest secure GMW protocol, a circuit that
   - reconstructs secret $s \in \{0,1\}^\ell$ using $\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}}$ on input $\langle s_i \rangle_B$, and
   - accepts $r_i \in \{0,1\}^{\gamma(N)}$ from each $P_i \in B$, where r_i was the value input by P_i in Step 1 above, and computes $r = \bigoplus_{P_i \in B} r_i$, and
   - computes output $z \leftarrow \mathcal{R}(s, r, a_1, \ldots, a_t)$, and
   - delivers output z to each party $P_j \in B$.

Since all communication between parties in B is via private pairwise point-to-point channels, and since the query generation algorithm and reconstruction algorithm are emulated by parties in B via semi-honest secure GMW protocol, it is clear that any coalition C of size less than t learns no more information about s other than what was revealed by the PIR queries $\{q_i\}_{i \in C}$ and the output z. Since $\mathcal{P}$ is a $(t-1)$-private PIR scheme, we thus conclude that any coalition of size less than t learns no more information about s other than what was implied by $f(s)$. Finally, since the GMW protocol [32, 33] involving t parties requires $O(t^2)$ pairwise random OT correlations to evaluate each AND gate in a circuit, it is easy to see that the total size of random OT correlations distributed by the dealer in the initialization phase is at most $O(n^2(\tau(\mathcal{Q}) + \tau(\mathcal{R}) + \tau(\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}})))$. □

It is easy to see that for the specific case of $n = 2$ and $t = 2$, we may use as $\mathcal{P}$ the 1-private 2-server PIR scheme of [20] (cf. Appendixapp2server). In this case, $\tau(\mathcal{Q}) = \widetilde{O}(2^{\ell/3})$ and $\tau(\mathcal{R}) = \widetilde{O}(2^{\ell/3})$, and since $\widetilde{O}(2^{\ell/3})$ dominates $\tau(\mathrm{Rec}^{n,t,\ell}_{\mathrm{Shamir}})$ when $\ell = \omega(\mathrm{polylog}(n))$, we conclude that the total share size of the above 2-out-of-2 secret-sharing scheme for $\mathcal{ALL}_\ell$ is $\widetilde{O}(2^{\ell/3})$.

H PSM with Universal Reconstruction for a Class of Functions

In this section, we prove a simple lower bound on the size of messages in a 2-party PSM protocol when the PSM referee's reconstruction function g is *universal* (for a class of functions) in that it does not depend on the specific function that is being computed. We present the formal definition below.

Definition H.1 (Private Simultaneous Messages with Universal Reconstruction for a Class of Functions)**.** *Let $X_1, \ldots, X_k, Z$ be finite domains, and let $X = X_1 \times \cdots \times X_k$. Let $\mathcal{F}$ denote the class of all functions from domain X to range Z. A private simultaneous messages (PSM) protocol $\mathcal{P}$ with* universal reconstruction for a class of functions *$\mathcal{F}$, computing any function $f \in \mathcal{F}$, consists of:*

- *A finite domain R of shared random inputs, and k finite message domains $M_1, \ldots, M_k$.*
- *For each $f \in \mathcal{F}_N$, message computation functions $\mu^f_1, \ldots, \mu^f_k$, where $\mu^f_i : X_i \times R \to M_i$.*
- *A* universal *reconstruction function that depends only on $\mathcal{F}$, denoted $g_{\mathcal{F}} : M_1 \times \cdots \times M_k \to Z$.*

Let $\mu^f(x, r)$ denote the k-tuple of messages $(\mu^f_1(x_1, r), \ldots, \mu^f_k(x_k, r))$. We say that the protocol $\mathcal{P}$ is correct *(with respect to $\mathcal{F}$), if for every $f \in \mathcal{F}$ and every input $x \in X$ and every random input $r \in R$, $g_{\mathcal{F}}(\mu^f(x, r)) = f(x)$. We say that the protocol $\mathcal{P}$ is* private *(with respect to $\mathcal{F}$), if for every $f \in \mathcal{F}$, the distribution of $\mu^f(x, r)$, where r is a uniformly random element of R, depends only on $f(x)$. That is, for every $f \in \mathcal{F}$ and every two inputs $x, x' \in X$ such that $f(x) = f(x')$, the random variables $\mu^f(x, r)$ and $\mu^f(x', r)$ (over a uniform choice of $r \in R$) are identically distributed. $\mathcal{P}$ is a PSM protocol with universal reconstruction for a class of functions $\mathcal{F}$ if it is both correct and private.*

The communication complexity of the PSM protocol $\mathcal{P}$ is naturally defined as $\sum_{i=1}^n \log |M_i|$. The randomness complexity of the PSM protocol $\mathcal{P}$ is defined as $\log |R|$.

The above definition differs from Definition 2.3 in that the PSM protocol is defined for an arbitrary class of functions $\mathcal{F}$, and that the referee's reconstruction function $g_{\mathcal{F}}$ is independent of the specific function $f \in \mathcal{F}$

being computed. An example of a PSM protocol with universal reconstruction for the class of functions $\mathcal{F}_N$ is the basic truth-table based PSM protocol of [30] (cf. Appendix E).

However, our construction of PSM protocols in Section 6 is *not* a PSM protocol with universal reconstruction for $\mathcal{F}_N$. Indeed, the referee's computation depends on the truth table of the specific $f \in \mathcal{F}_N$ being computed. (Note that in our construction in Figure 3 (see also Section 6.3, the referee computes, for a server holding as database the truth-table of f, the server's answer to a decomposable PIR query.) Another example of a scheme in the literature that is not a PSM protocol with universal reconstruction for $\mathcal{F}_N$ is the (information-theoretic) Yao garbled circuit based PSM protocol of [30].

While the PSM protocols (*without* universal reconstruction) that we constructed in Section 6 have communication complexity $O(N^{1/2})$, all known PSM protocols *with* universal reconstruction for $\mathcal{F}_N$ have communication complexity $O(N)$. Our next theorem shows that this situation is somewhat inherent for PSM protocols with universal reconstruction for the class of functions $\mathcal{F}_N$.

Theorem H.2. *Let $\mathcal{P}$ be a 2-party PSM protocol with universal reconstruction for function class $\mathcal{F}_N$. Then, the communication complexity of $\mathcal{P}$ is at least N.*

Proof. Let the domain of messages for party P_1 and P_2 be M_1 and M_2 respectively. Let the message computation function for party P_1 and P_2 for $f \in \mathcal{F}_N$ be μ_1^f and μ_2^f respectively. Consider an execution of the PSM protocol where the shared random input $r \in R$, where R is the finite domain of shared random inputs, is fixed. Now for $f \in \mathcal{F}_N$, consider the set of values $M^f = \{\mu_1^f(x_1, r)\}_{x_1 \in [N]} \cup \{\mu_2^f(x_2, r)\}_{x_2 \in [N]}$. Since the referee's reconstruction procedure is universal, i.e., consists of a single function $g_{\mathcal{F}_N}$ for every $f \in \mathcal{F}_N$, and the PSM protocol is perfectly correct, it is easy to see that for each $f \in \mathcal{F}_N$, the set M^f alone must completely define the values of $\{f(x_1, x_2)\}_{(x_1,x_2) \in [N] \times [N]}$, i.e., the function f. That is, the size of M^f, i.e., $N \cdot (\log |M_1| + \log |M_2|)$, must be greater than or equal to the number of bits required to define $f \in \mathcal{F}_N$, i.e., $\log(2^{N^2}) = N^2$. Since the total length of the messages in a given PSM execution is equals $\log |M_1| + \log |M_2|$, we conclude that the communication complexity of an execution of the PSM protocol with universal reconstruction is least N. □

I Private Simultaneous Messages – Additional details

In this appendix, we provide some missing details from Section 6.

We start by a more formal description of the protocol from Section 6.3, for which we will also add some additional notation. Motivated by the application, the database length is now N^2-bit and the obtained complexity will hence be $O(N^{1/2})$. The truth table of $f : [N] \times [N] \to \{0,1\}$ is naturally associated with the 4-dimensional cube $[N^{1/2}]^4$, where each position $x = (x_1, x_2) \in [N] \times [N]$ in the truth-table is associated with a 4-tuple $(\hat{x}_1, \ldots, \hat{x}_4) \in [N^{1/2}]^4$, in a natural manner. Specifically, we associate $(\hat{x}_1, \hat{x}_2)$ with x_1, and $(\hat{x}_3, \hat{x}_4)$ with x_2. We also use $g(\hat{x}_1, \ldots, \hat{x}_4)$ to denote the reverse mapping from the 4-dimensional cube element $(\hat{x}_1, \ldots, \hat{x}_4)$ to the corresponding element (x_1, x_2) in the database.

The PIR algorithms are defined below. Let the client's input be $x = (x_1, x_2) \in [N] \times [N]$. As described above, let $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4)$ be the position in the 4-dimensional cube that is associated with x. For any $\hat{x} \in [N^{1/2}]$, let $\mathbf{e}_{\hat{x}}$ denote the $N^{1/2}$-bit indicator vector which has 1 at location $\hat{x}$, and 0 elsewhere. Finally, for any 4-tuple of strings of length $N^{1/2}$, define a function $G_{\mathrm{CGKS}} : \{0,1\}^{N^{1/2}} \times \{0,1\}^{N^{1/2}} \times \{0,1\}^{N^{1/2}} \times \{0,1\}^{N^{1/2}} \to \{0,1\}$ as $G_{\mathrm{CGKS}}(j_1, j_2, j_3, j_4) = \bigoplus_{i_1,i_2,i_3,i_4 \in [N^{1/2}]} (\langle j_1, \mathbf{e}_{i_1} \rangle \cdot \langle j_2, \mathbf{e}_{i_2} \rangle \cdot \langle j_3, \mathbf{e}_{i_3} \rangle \cdot \langle j_4, \mathbf{e}_{i_4} \rangle \cdot g(i_1, i_2, i_3, i_4))$, where g is the mapping defined above. That is, $G_{\mathrm{CGKS}}(j_1, j_2, j_3, j_4)$ is the exclusive-or of all locations (i_1, i_2, i_3, i_4) in the 4-dimensional cube such that the i_k-th bit of j_k equals 1 for all $1 \le k \le 4$.

- *Decomposable query generation.* Let $u = (\hat{u}_1^0, \hat{u}_2^0, \hat{u}_3^0, \hat{u}_4^0)$ where each of $\hat{u}_1^0$, $\hat{u}_2^0$, $\hat{u}_3^0$, $\hat{u}_4^0$ are uniformly random strings of bit length $N^{1/2}$.
  - Algorithm $\mathcal{Q}_1(x_1, u)$: Output $q_1 = (x_1, u)$.
  - Algorithm $\mathcal{Q}_2(x_2, u)$: Output $q_2 = (x_2, u)$.
  - Algorithm $\mathcal{Q}_3(x_1, x_2, u)$: Recall that x_1 is associated with the tuple $(\hat{x}_1, \hat{x}_2)$, and x_2 is associated with the tuple $(\hat{x}_3, \hat{x}_4)$.
    - Algorithm $\mathcal{Q}_3^1(x_1, u)$: Output $q_3^1 = (\hat{u}_1^0 \oplus \mathbf{e}_{\hat{x}_1}, \hat{u}_2^0 \oplus \mathbf{e}_{\hat{x}_2})$.
    - Algorithm $\mathcal{Q}_3^2(x_2, u)$: Output $q_3^2 = (\hat{u}_3^0 \oplus \mathbf{e}_{\hat{x}_3}, \hat{u}_4^0 \oplus \mathbf{e}_{\hat{x}_4})$.
- *Answering algorithm.*

– Algorithm $\mathcal{A}(1, q_1 = (x_1, u), f)$: Recall x_1 is associated with the tuple $(\hat{x}_1, \hat{x}_2)$. Define $\hat{u}_1^1 = \mathbf{e}_{\hat{x}_1} \oplus \hat{u}_1^0$ and $\hat{u}_2^1 = \mathbf{e}_{\hat{x}_2} \oplus \hat{u}_2^0$. Initialize set $A_1 = \emptyset$. Do:
  * For $b \in \{0000, 0100, 1000, 1100\}$, with $b = b_1\|b_2\|b_3\|b_4$, compute $\tilde{a}_b = G_{\mathrm{CGKS}}(\hat{u}_1^{b_1}, \hat{u}_2^{b_2}, \hat{u}_3^{b_3}, \hat{u}_4^{b_4})$, and add $\tilde{a}_b$ to A_1.
  * For $b \in \{0101, 1001, 1101\}$, with $b = b_1\|b_2\|b_3\|b_4$, and for $\ell \in [N^{1/2}]$, compute $\tilde{a}_b^\ell = G_{\mathrm{CGKS}}(\hat{u}_1^{b_1}, \hat{u}_2^{b_2}, \hat{u}_3^{b_3}, \hat{u}_4^{b_4} \oplus \mathbf{e}_\ell)$, and add $\tilde{a}_b^\ell$ to A_1.
  * For $b \in \{0110, 1010, 1110\}$, with $b = b_1\|b_2\|b_3\|b_4$, and for $\ell \in [N^{1/2}]$, compute $\tilde{a}_b^\ell = G_{\mathrm{CGKS}}(\hat{u}_1^{b_1}, \hat{u}_2^{b_2}, \hat{u}_3^{b_3} \oplus \mathbf{e}_\ell, \hat{u}_4^{b_4})$, and add $\tilde{a}_b^\ell$ to A_1.

  Set $a_1 = A_1$, and output a_1.
– Algorithm $\mathcal{A}(2, q_2 = (x_2, u), f)$: Recall x_2 is associated with the tuple $(\hat{x}_3, \hat{x}_4)$. Define $\hat{u}_3^1 = \mathbf{e}_{\hat{x}_3} \oplus \hat{u}_3^0$ and $\hat{u}_4^1 = \mathbf{e}_{\hat{x}_4} \oplus \hat{u}_4^0$. Initialize set $A_2 = \emptyset$. Do:
  * For $b \in \{0001, 0010, 0011\}$, with $b = b_1\|b_2\|b_3\|b_4$, compute $\tilde{a}_b = G_{\mathrm{CGKS}}(\hat{u}_1^{b_1}, \hat{u}_2^{b_2}, \hat{u}_3^{b_3}, \hat{u}_4^{b_4})$, and add $\tilde{a}_b$ to A_2.
  * For $b \in \{0111\}$, with $b = b_1\|b_2\|b_3\|b_4$, and for $\ell \in [N^{1/2}]$, compute $\tilde{a}_b^\ell = G_{\mathrm{CGKS}}(\hat{u}_1^{b_1}, \hat{u}_2^{b_2} \oplus \mathbf{e}_\ell, \hat{u}_3^{b_3}, \hat{u}_4^{b_4})$, and add $\tilde{a}_b^\ell$ to A_2.
  * For $b \in \{1011\}$, with $b = b_1\|b_2\|b_3\|b_4$, and for $\ell \in [N^{1/2}]$, compute $\tilde{a}_b^\ell = G_{\mathrm{CGKS}}(\hat{u}_1^{b_1} \oplus \mathbf{e}_\ell, \hat{u}_2^{b_2}, \hat{u}_3^{b_3}, \hat{u}_4^{b_4})$, and add $\tilde{a}_b^\ell$ to A_2.

  Set $a_2 = A_2$, and output a_2.
– Algorithm $\mathcal{A}_3(3, q_3 = (q_3^1, q_3^2), f)$: Parse q_3^1 as $(\hat{u}_1^1, \hat{u}_2^1)$, and q_3^2 as $(\hat{u}_3^1, \hat{u}_4^1)$. Compute $\tilde{a}_{1111} = G_{\mathrm{CGKS}}(\hat{u}_1^1, \hat{u}_2^1, \hat{u}_3^1, \hat{u}_4^1)$. Set $a_3 = \tilde{a}_{1111}$, and output a_3.

- *Decomposable reconstruction algorithm.*
  – Algorithm $\mathcal{R}'((x_1, x_2), r, a_1, a_2)$: Recall x_1 is associated with the tuple $(\hat{x}_1, \hat{x}_2)$, and x_2 is associated with the tuple $(\hat{x}_3, \hat{x}_4)$.
    * Compute $\tilde{z} = \left(\bigoplus_{b \in \{0000,0100,1000,1100\}} \tilde{a}_b\right) \oplus \left(\bigoplus_{b \in \{0001,0010,0011\}} \tilde{a}_b\right)$.
    * Set $\tilde{z}_1 = \tilde{a}_{1011}^{\hat{x}_1}$, and $\tilde{z}_2 = \tilde{a}_{0111}^{\hat{x}_2}$.
    * Compute $\tilde{z}_3 = \bigoplus_{b \in \{0110,1010,1110\}} \tilde{a}_b^{\hat{x}_3}$.
    * Compute $\tilde{z}_4 = \bigoplus_{b \in \{0101,1001,1101\}} \tilde{a}_b^{\hat{x}_4}$.
    * Output $z' = \tilde{z} \oplus \tilde{z}_1 \oplus \tilde{z}_2 \oplus \tilde{z}_3 \oplus \tilde{z}_4$.
  – Algorithm $\mathcal{R}''(a_3, z')$: Output $z = a_3 \oplus z'$.

Efficiency of the resulting PSM protocol in Figure 3. First observe that the length of each of r, a_1, a_2,q_3^1,q_3^2 in the 3-server decomposable PIR protocol described above is $O(N^{1/2})$. Next, in algorithm $\mathcal{R}'$, each of the values $\tilde{z}_1, \tilde{z}_2, \tilde{z}_3, \tilde{z}_4$ can be computed by a multiplicative depth-1 circuit of size $O(N^{1/2})$. Given this, we conclude that there exists a PSM protocol $\pi_{\mathcal{R}'}$ (e.g., based on information theoretic Yao garbled circuits) whose communication (and randomness) complexity is $O(N^{1/2})$. Thus $|m_1| + |m_2| + |q_3^1| + |q_3^2| = O(N^{1/2})$, and so the communication complexity is $O(N^{1/2})$. The randomness complexity of $\pi_{\mathcal{R}'}$ is $O(N^{1/2})$ and so the randomness complexity of the resulting PSM protocol is also $O(N^{1/2})$.

I.1 Decomposable Matching Vectors and Decomposable PIR Schemes

Motivated by the possibility of constructing decomposable 3-server PIR protocols that are as efficient as the best known 3-server PIR protocols of [61, 27, 9] (whose communication complexity is subexponential in $\log N$), we attempt to construct decomposable 3-server PIR protocols using (appropriate variants of) the machinery employed in these 3-server PIR protocols. This machinery includes share conversion (cf. Section A.1.1), and the so-called *matching vectors family* which we define below.

Definition I.1 (Matching vectors family [61, 27])**.** *Let $S \subseteq \mathbb{Z}_m \setminus \{0\}$. We say that $U = \{u_x\}_{x \in [N] \times [N]}$ and $V = \{v_x\}_{x \in [N] \times [N]}$ in $\mathbb{Z}_m^h$ form an S-matching family of size N^2 if the following conditions hold: (1) $\langle u_x, v_x \rangle = 0$ for every $x \in [N] \times [N]$, and (2) $\langle u_x, v_{x'} \rangle \in S$ for every $x \neq x'$.*

For our purposes we will need a variant of matching vectors family Unfortunately, we do not know how to construct decomposable matching vectors family, and leave it as an open question to find decomposable matching vectors family such that h is subexponential in $\log N$, or even asymptotically better than $N^{1/2}$.

Definition I.2 (Decomposable matching vectors family)**.** *Let $S \subseteq \mathbb{Z}_m \setminus \{0\}$. We say that $U^{(1)} = \{u^{(1)}_{x_1}\}_{x_1 \in [N]}$, $U^{(2)} = \{u^{(2)}_{x_2}\}_{x_2 \in [N]}$, and $V = \{v_x\}_{x \in [N] \times [N]}$, in $\mathbb{Z}^h_m$ form a* decomposable *S-matching family of size N^2 if the following conditions hold: (1) $\langle u^{(1)}_{x_1} + u^{(2)}_{x_2}, v_x \rangle = 0$, for every $x = (x_1, x_2) \in [N] \times [N]$, and (2) $\langle u^{(1)}_{x_1} + u^{(2)}_{x_2}, v_x \rangle \in S$, for every x such that $x \neq (x_1, x_2)$.*

Let $U^{(1)}, U^{(2)}, V$ be a decomposable S-matching family for $S = S_m$. Further let m, p, β be such that there exists a share conversion from 3-party CNF sharing over $\mathbb{Z}_m$ to additive sharing over $\mathbb{Z}^\beta_p$ with respect to C_{S_m} (cf. Fact A.7). Given the above, we show how to construct a decomposable 3-server PIR protocol.

- *Decomposable query generation algorithm.* Let the matching vectors corresponding to the client's input, say $x = (x_1, x_2) \in [N] \times [N]$, be $(u^{(1)}_{x_1}, u^{(2)}_{x_2}) \in \mathbb{Z}^h_m \times \mathbb{Z}^h_m$. Let the client's randomness be $r = (r_1, r_2)$, where r_1, r_2 are interpreted as vectors in $\mathbb{Z}^h_m$. Then (1) algorithm $\mathcal{Q}_1(x_1, r)$ outputs $q_1 = (u^{(1)}_{x_1} - r_1, r_1 + r_2)$, (2) algorithm $\mathcal{Q}_2(x_2, r)$ outputs $q_2 = (r_1 + r_2, u^{(2)}_{x_2} - r_2)$, and (3) algorithm $\mathcal{Q}^1_3(x_1, r)$ outputs $q^1_3 = u^{(1)}_{x_1} - r_1$, and $\mathcal{Q}^2_3(x_2, r)$ outputs $q^2_3 = u^{(2)}_{x_2} - r_2$, i.e., algorithm $\mathcal{Q}_3(x, r)$ outputs $(u^{(1)}_{x_1} - r_1, u^{(2)}_{x_2} - r_2)$.

  Observe that the matching vector $(u^{(1)}_{x_1} + u^{(2)}_{x_2})$ corresponding to input $x = (x_1, x_2)$ is effectively 3-CNF shared among the 3 servers by the decomposable query generation algorithm. To see this, denote $u'_1 = u^{(2)}_{x_2} - r_2$, $u'_2 = u^{(1)}_{x_1} - r_1$, and $u'_3 = r_1 + r_2$. Then, $\{u'_2, u'_3\}$ can be derived from q_1, and $\{u'_3, u'_1\}$ can be derived from q_2, and $\{u'_1, u'_2\}$ can be derived from $q_3 = (q^1_3, q^2_3)$. It is easy to verify that $\{u'_2, u'_3\}, \{u'_3, u'_1\}, \{u'_1, u'_2\}$ is a 3-party CNF sharing of $u^{(1)}_{x_1} + u^{(2)}_{x_2}$.
- *Answering algorithm.* The answering algorithm is similar to the answering algorithms of the 3-server PIR protocols of [27, 9]. Each server $\mathcal{S}_j$ performs the following:
  - $\mathcal{S}_j$ derives $\{u'_{j-1}, u'_{j+1}\}$ from q_j as described above.
  - $\mathcal{S}_j$ computes $a_{j,j-1,\ell} = \langle u'_{j-1}, v_\ell \rangle$ and $a_{j,j+1,\ell} = \langle u'_{j+1}, v_\ell \rangle$ for each $\ell \in [N] \times [N]$.
  - $\mathcal{S}_j$ applies the share conversion procedure (corresponding to share conversion from 3-party CNF sharing to additive sharing over $\mathbb{Z}^\beta_p$) to each CNF share $\{a_{j,j-1,\ell}, a_{j,j+1,\ell}\} \in \mathbb{Z}^2_m$ to obtain $a'_{j,\ell} \in \mathbb{Z}^\beta_p$.
  - Finally, $\mathcal{S}_j$ sends $a_j = \sum_{\ell \in [N] \times [N]} y_\ell \cdot a'_{j,\ell}$ to the client, where $y_\ell \in \{0, 1\}$ denotes the ℓth element of the database (i.e., $y_\ell = f(\ell)$), and is interpreted as an element in $\mathbb{Z}^\beta_p$.
- *Decomposable reconstruction algorithm.* Let $\mathcal{R}'((x_1, x_2), r, a_1, a_2)$ be an algorithm that computes $a_1 + a_2$, where addition is over $\mathbb{Z}^\beta_p$. Let $\mathcal{R}''(c_1, c_2)$ be an algorithm that outputs 0 if $c_1 + c_2 = 0 \in \mathbb{Z}^\beta_p$, and outputs 1 if $c_1 + c_2 \neq 0$. The client obtains answers a_1, a_2, a_3 from the servers, and computes $a = a_3 + \mathcal{R}'((x_1, x_2), r, a_1, a_2) = a_1 + a_2 + a_3 \in \mathbb{Z}^\beta_p$, and outputs $z = \mathcal{R}''(a)$.

Efficiency of the resulting PSM protocol. We consider the setting where values m, p, β are constants independent of N. In this case, it is easy to see that the PSM messages for realizing $\mathcal{R}'$ will be of length $O(1)$. On the other hand, the lengths of $q^1_3, q^2_3 \in \mathbb{Z}^h_m$ will be $O(h)$ (once again assuming m is a constant), and will dominate the PSM cost. Finally, we note that the randomness complexity of the PSM protocol is also $O(h)$.

Unfortunately, we do not know how to construct decomposable matching vectors family, and leave it as an open question to find decomposable matching vectors family such that h is subexponential in $\log N$, or even asymptotically better than $N^{1/2}$.

J Secret Sharing Schemes for Forbidden Graph Access Structures

In this section, we construct secret-sharing schemes realizing $\mathcal{A}^G$ for general graphs, using the secret-sharing scheme for forbidden bipartite graph access structures, presented in Section 7.

Theorem J.1. *Let $G = (V, E)$ be a graph where $|V| = N$. There exists a secret sharing realizing $\mathcal{A}^G$ with domain of secrets $\{0, 1\}$ and total share size $O(N^{3/2} \log N)$.*

Proof. To construct a scheme realizing $\mathcal{A}^G$, we rely on the fact that an arbitrary graph $G = (V, E)$, with $|V| = N$, can be covered by $\log N$ graphs $G_1, \ldots, G_{\log N}$, such that (1) for every i, the graph G_i is a bipartite subgraph of G, and (2) an edge $(u, v) \in E$ iff there exists some $j \in [\log N]$ such that (u, v) is an edge in G_j. Such a cover can, for instance, be obtained by defining L_j as the set of all vertices in V whose label contains

0 in its jth bit, R_j as the remaining set of vertices in V whose label contains 1 in its jth bit, and having an edge $(x, y) \in E_j$ if and only if $x \in L_j, y \in R_j$, and $(x, y) \in E$. As for every $(x, y) \in E$ (where $x \neq y$) there exists at least one bit where x and y differ, then (x, y) is an edge in at least one G_j.

We claim that $\mathcal{A}^G$ is equal to $\cap_{i=1}^{\log N} \mathcal{A}^{G_i}$. On one hand, a set is authorized in $\mathcal{A}^G$ if it is of size at least 3 or it is a set $\{x, y\}$ such that $(x, y) \notin E$ and, therefore, $(x, y) \notin E_j$ for every j. On the other hand, if a set $\{x, y\}$ is not authorized in $\mathcal{A}^G$, then $(x, y) \in E_j$ for some j and, thus, not in $\mathcal{A}^{G_i}$.

We next describe a secret-sharing scheme realizing $\mathcal{A}^G$. The dealer, with input $s \in \{0, 1\}$, picks at random $s_1, \ldots, s_{\log N}$ such that $s = s_1 \oplus \cdots \oplus s_{\log N}$, and shares each s_i with a scheme realizing $\mathcal{A}^{G_i}$ with total share size $O(N^{3/2})$ (such scheme exists by Corollary 7.4). For the correctness and privacy, note that a set A can reconstruct s if and only if A can reconstruct every s_i, which happens if and only if $A \in \mathcal{A}^{G_i}$, for every i, which is equivalent to $A \in \mathcal{A}^G$. □

Remark J.2. We can also construct a secret sharing realizing $\mathcal{A}^G$ with domain of secrets $\{0, 1\}^{O(\log N)}$ and total share size $O(N^{3/2} \log N)$, that is, the information ratio of the scheme (the ratio between the length of the shares and the length of the secret) is improved by a factor of $\log N$. To achieve this, we start with a cover of G with $O(\log N)$ bipartite graphs $G_1, \ldots, G_{O(\log N)}$ such that every edge in G is an edge in at least a constant fraction β of the the bipartite graphs (for example, rename the vertices in G to random numbers in $[N^2]$ and use the above construction with the new names). On one hand, if a set is authorized in $\mathcal{A}^G$, then it is authorized in $\mathcal{A}^{G_i}$ for every i. On the other hand, if a set is unauthorized in $\mathcal{A}^G$, then it is unauthorized in $\mathcal{A}^{G_i}$ for at least a constant fraction β of the access structures. We can, thus, use the decomposition construction of Stinson [57] to get the desired scheme.

K Proofs

In this section, we provide formal proofs of Theorems 3.1 and 3.3 stated in Section 3.

K.1 Secure Computation in the OT-hybrid Model

Proof of Theorem 3.1. The protocol of Figure 1, denoted π, is the desired protocol. We will prove that this protocol is information theoretically secure in the OT-hybrid model and has the desired efficiency.

- *Privacy.* First, note that in the OT-hybrid model (alternatively, OT-preprocessing model), the 2-party GMW protocol is information-theoretically secure against passive adversaries [32, 33, 3]. Given this, the security of the protocol of Figure 1 can be reduced information theoretically to the security of a protocol π' in which instead of running the GMW protocol in Step 1 (resp. Step 3), parties simply provide their input to the GMW protocol for evaluating $\widetilde{\mathcal{Q}}$ (resp. $\widetilde{\mathcal{R}}$) to an ideal functionality that directly evaluates $\widetilde{\mathcal{Q}}$ (resp. $\widetilde{\mathcal{R}}$) over the provided inputs and delivers to the parties their corresponding output. It is easy to see that in protocol π' the view of the passively corrupted party P_i equals its input x_i, randomness $r^{(i)}$, a single PIR query q_i, and the final output z. First, note that the value $r^{(i)}$ is information theoretically independent of the randomness used in the algorithm $\widetilde{\mathcal{Q}}$. Next, note by the client privacy of the PIR scheme $\mathcal{P}$, the distribution of query q_i is independent of $x = (x_1, x_2)$, and in particular, is independent of the input x_{3-i} of the honest party P_{3-i}. Given the above, it is easy to see that the corrupt party learns no information about the honest party's input (i.e., x_{3-i}) other than what is revealed by the value $z = f(x_1, x_2)$.
- *Correctness.* Correctness follows immediately from the correctness of the 2-server PIR protocol $\mathcal{P}$ and the correctness of the GMW protocol (employed in Steps 1 and 3).
- *Efficiency.* From the description of the protocol π, we have that the GMW protocol is used to securely evaluate $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$. To evaluate a circuit containing s_1 AND gates and s_2 XOR gates, the GMW protocol requires $O(s_1)$ calls to an OT oracle, and has communication complexity $O(s_1 + s_2)$ [33]. Since $\tau(\widetilde{\mathcal{Q}}) = \tau(\mathcal{Q})$, and $\tau(\widetilde{\mathcal{R}}) = \tau(\mathcal{R})$ hold ($C(\widetilde{\mathcal{Q}})$ (resp. $C(\widetilde{\mathcal{R}})$) has the same number of AND gates as $C(\mathcal{Q})$ (resp. $C(\mathcal{R})$) and differs in the number of XOR gates), and since the number of XOR gates is bounded, up to constant factors, by the number of bits of the random input to circuits $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$, we conclude that the communication complexity (including calls to an OT oracle) of the protocol of Figure 1 is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$ bits.

□

K.2 Secure Computation in the Preprocessing Model

Proof of Theorem 3.3. The protocol of Figure 2, denoted π, is the desired protocol. We will prove that this protocol is information theoretically secure in the preprocessing model and has the desired efficiency.

- *Privacy.* First, note that given precomputed oblivious transfers, the 2-party GMW protocol is information-theoretically secure against passive adversaries [32, 33, 3]. Given this, the security of the protocol of Figure 2 can be reduced information theoretically to the security of a protocol π' in which instead of running the GMW protocol in Step 1 (resp. Step 3), parties simply provide their input to the GMW protocol for evaluating $\widetilde{\mathcal{Q}}$ (resp. $\widetilde{\mathcal{R}}$) to an ideal functionality that directly evaluates $\widetilde{\mathcal{Q}}$ (resp. $\widetilde{\mathcal{R}}$) over the provided inputs and delivers to the parties their corresponding output. It is easy to see that in protocol π' the view of the passively corrupted party P_i equals its input x_i, randomness $r^{(i)}$, a random share $a_1^{(i)}$ (of $a_1 = \mathcal{A}(1, q_1, f)$), a single PIR query q_{i+1}, and the final output z. First, note that the value $r^{(i)}$ is information theoretically independent of the randomness used in the algorithm $\widetilde{\mathcal{Q}}$. Similarly, the value $a_1^{(i)}$ is information theoretically independent of the value a_1 and also of the input of the honest party P_{3-i} (i.e., x_{3-i}). Next, note by the client privacy of the PIR scheme $\mathcal{P}$, the distribution of query q_{i+1} is independent of $x = (x_1, x_2)$, and in particular, is independent of the input x_{3-i} of the honest party P_{3-i}. Given the above, it is easy to see that the corrupt party learns no information about the honest party's input (i.e., x_{3-i}) other than what is revealed by the value $z = f(x_1, x_2)$.
- *Correctness.* Correctness follows immediately from the correctness of the 3-server PIR protocol $\mathcal{P}$ and the correctness of the GMW protocol (employed in Steps 1 and 3).
- *Efficiency.* From the description of the protocol π, we have that the GMW protocol is used to securely evaluate $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$. To evaluate a circuit containing s_1 AND gates and s_2 XOR gates, the GMW protocol requires (random) OT correlations of size $O(s_1)$ [3], and has communication complexity $O(s_1 + s_2)$ [33]. Since $\tau(\widetilde{\mathcal{Q}}) = \tau(\mathcal{Q}_{-1}) \leq \tau(\mathcal{Q})$, and $\tau(\widetilde{\mathcal{R}}) = \tau(\mathcal{R})$ hold ($C(\widetilde{\mathcal{Q}})$ (resp. $C(\widetilde{\mathcal{R}})$) has the same number of AND gates as $C(\mathcal{Q}_{-1})$ (resp. $C(\mathcal{R})$) and differs in the number of XOR gates), and since the number of XOR gates in $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$ is bounded, up to constant factors, by the number of bits of the random input (in circuits $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$) and by the length of input a_1 (in circuit $C(\widetilde{\mathcal{R}})$), we conclude that the communication complexity of the protocol of Figure 2 is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$ bits. It is easy to see that the size of the random OT correlations required is indeed bounded by $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$ (which is in fact the number of AND gates in circuits $C(\widetilde{\mathcal{Q}})$ and $C(\widetilde{\mathcal{R}})$). Finally note that the lengths of $r^{(1)}$ and $r^{(2)}$ are bounded, up to constant factors, by $\tau(\mathcal{Q})$ (since $r = r^{(1)} \oplus r^{(2)}$ is an input to $C(\mathcal{Q})$), and that the lengths of $a_1^{(1)}$ and $a_1^{(2)}$ are bounded, up to constant factors, by $\tau(\mathcal{R})$ (since $a_1 = a_1^{(1)} \oplus a_1^{(2)}$ is an input to $C(\mathcal{R})$). Thus, we conclude that the correlated randomness complexity (including the size of the random OT correlations) is $O(\tau(\mathcal{Q}) + \tau(\mathcal{R}))$ bits.

□