On the Cryptographic Complexity of the Worst Functions*

Amos Beimel[1], Yuval Ishai[2], Ranjit Kumaresan[2], and Eyal Kushilevitz[2]

[1] Dept. of Computer Science, Ben Gurion University of the Negev, Be'er Sheva, Israel
amos.beimel@gmail.com
[2] Department of Computer Science, Technion, Haifa, Israel
{yuvali|ranjit|eyalk}@cs.technion.ac.il

Abstract. We study the complexity of realizing the "worst" functions in several standard models of information-theoretic cryptography. 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 $\widetilde{O}(\sqrt{N})$.

The worst-case complexity of the best previous solutions was $\Omega(N)$ for the first three problems and $\Omega(N/\log N)$ for the last one. 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.

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 $\Theta(N/\log N)$ [53, 49] and

* Research by the first three authors received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC. The fourth author was supported by ISF grant 1361/10 and BSF grant 2008411.

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 [48].[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 [5, 40] (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 [52, 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 [45, 46], and the possibility of efficiently precomputing [4] and (in the computational setting) extending [3, 37] OTs. See [43] 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, 3, 56]), and a more systematic study was conducted in [12, 51].

[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.

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 [17], we get an upper bound of $N - 1$ on the OT complexity of the worst-case f. This raises the following question:

Question 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, 42], the answer to Question 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 [12].

We will also be interested in a sampling variant of Question 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 captures the rate of securely reducing complex correlations to simple ones, a question which was recently studied in [51].

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 [4]. The model is motivated by the possibility of generating the correlated randomness using semi-trusted servers or a (computationally) secure interactive protocol, thus capturing the goal of minimizing the online complexity of secure computation via offline preprocessing. See [14, 41, 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

[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 [20].

from [41], 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 the 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 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 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

[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.

graph-based access structures was considered in [54] 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 $O(N/\log N)$ [18, 16, 28]. This raises the following question:

Question 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 an improved 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 [9]. 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 [21]. See full version for more details.

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, 41] 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 [21]. In the hope of improving our $O(\sqrt{N})$ upper bound, we reduce the problem of decomposable 3-server PIR to a combinatorial question of obtaining a decomposable variant of matching vector sets [58, 26]. See full version for more details. 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 [50] 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 [7] 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 was known is that of secure multiparty computation in the presence of an honest majority. While the complexity of standard protocols [13, 19] 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. [5] 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 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) [21]. 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 [40] 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 [40] is completely general. On the down side, it does not apply in the two-party case and it requires (information theoretic) PIR protocols with polylogarithmic communication, which are not known to exist for a constant number of servers k.

Beimel and Malkin [12] 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 [44]. Prabhakaran and Prabhakaran [51] put forward the question of characterizing the rate of secure reductions between *sampling* functionalities, and strengthen previous negative results from [56] 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 [56] 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 [54], 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 (linear) scheme in which the share size of each party is $O(N/\log N)$ [18, 16, 28]. The best lower bound for the total share size required to realize a graph access structure by a general secret-sharing scheme is $\Omega(N \log N)$ [55, 15, 23]. The best lower bound for total share size required to realize a graph access structure by a linear secret-sharing scheme is $\Omega(N^{3/2})$ [7]. The problem of secret sharing for dense graphs was studied in [8]. Additional references on secret sharing of graph access structures can be found in [8].

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 (measured as number of AND gates) of a boolean circuit, over the basis $\{\wedge, \oplus, \neg\}$, that represents $\mathcal{B}$.

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 mainly to the case of semi-honest (passive) corruptions. (In Appendix B, we show how to extend some of our results to the malicious setting.) 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., [41, 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 1 (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 $j \in [k]$, 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 q 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., $\sum_j (|q_j| + |a_j|) = k(\alpha(N) + \beta(N))$).*

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 $\mathcal{Q}_1, \mathcal{Q}_{-1}$ and 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 (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 $\log |R|$.

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 1. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 2-server PIR scheme for $\mathcal{F}_N$ as described in Definition 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 [21] in Theorem 1, we obtain:

Corollary 1. *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 [17].

Secure computation in the preprocessing model. Since OTs can be precomputed [4], the protocol implied by Theorem 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 2. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 3-server PIR scheme for $\mathcal{F}_N$ as described in Definition 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 1. 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 [41]. 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 2, 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, 11] in Theorem 2, we obtain:

Corollary 2. *For any 2-party functionality* $f : [N] \times [N] \rightarrow \{0,1\}$*, 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* $2^{\widetilde{O}(\sqrt{\log N})}$.

While we mainly focus here on efficiency of 2-party secure computation, we show how to construct protocols in the multiparty setting, and also for the setting with honest majority in Appendix A. We summarize our results on t-private k-party semihonest secure computation in Table 1. In Appendix B we show how to extend our results on secure computation to the malicious setting.

Complexity measure	(t,k)	This work	Reference
OT complexity in the OT-hybrid model	$(1,2)$	$O(N^{2/3})$	Cor. 1
	$(t, k \leq 2t)$	$N^{k/\lfloor 2k-1/t \rfloor} \cdot \mathrm{poly}(k)$	Cor. 5
Correlated randomness complexity in the preprocessing model	$(1,2)$	$2^{\widetilde{O}(\sqrt{\log N})}$	Cor. 2
	$(t > 1, k \leq 2t)$	$N^{k/\lfloor 2k+1/t \rfloor} \cdot \mathrm{poly}(k)$	Cor. 4
Communication complexity in the plain model	$(t, 2t < k < 3^t)$	$N^{k/\lfloor 2k-1/t \rfloor} \cdot \mathrm{poly}(k)$	Cor. 4
	$(t, k \geq 3^t)$	$2^{\widetilde{O}(\sqrt{k \log N})} \cdot \mathrm{poly}(k)$	Cor. 5

Table 1. Summary of upper bounds on different complexity measures of t-private k-party semihonest secure computation of the worst function $f : [N]^k \rightarrow \{0,1\}$.

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

Theorem 3. *For any 2-party functionality* $f : [N] \times [N] \rightarrow \{0,1\}$*, 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 randomness 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 [54] 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 4. *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 $O(N^{3/2} \log N)$.*

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 has 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 $\Omega(N)$ [17, 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 full version.

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. [21] 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. This proves Corollary 1.

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.

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

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 $\Omega(N)$ [17, 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. 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. Instantiating the protocol described above with the best known 3-server PIR protocol [58, 27, 11] we obtain a perfectly secure protocol in the preprocessing model whose communication and correlated randomness complexity is $2^{\widetilde{O}(\sqrt{\log N})}$. The details are deferred to the full version.

6 Private Simultaneous Messages

In this section, we provide a new framework for constructing PSM protocols (cf. Definition 2). 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 1) 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 3 (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 $\mathcal{R}$. 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. *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. *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 q_k, the query 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. We 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 π for computing $\mathcal{R}'$. (We remark that $\mathcal{R}'$ is "simpler" than f, and consequently existing PSM protocols (e.g., [38, 47]) can realize $\mathcal{R}'$ very efficiently.) In the protocol, P_1, holding x_1 and 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 and 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 above 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 1. *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 π' 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 π 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 [21]. We start with a high level description of this approach, specifically for the case of 4-dimensional cubes and of its adaptation to the decomposable case. In the following, for set S and element i, let $S \oplus \{i\}$ denote the set $S \setminus \{i\}$ if $i \in S$, and $S \cup \{i\}$ otherwise.

The starting point of the CGKS [21] 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 client could send each query $(T_1, \ldots, T_4)$ to a different server ($4 \cdot n^{1/4}$ bits to each), who would 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 [21] 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 needed 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 the 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 full version). 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 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 $O(N^2/\log N)$ [18, 16, 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. For lack of space, our generalized construction is presented in the full version. We start by formally defining forbidden graph access structures.

Definition 4. *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 5. *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 2. *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). 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). 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). $\square$

Using the PSM protocols described in Theorem 3 in Lemma 2, we get the following corollary.

Corollary 3. *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 the full version, we show how to construct secret-sharing schemes realizing $\mathcal{A}^G$ for general graphs, using the secret-sharing scheme for forbidden bipartite graph access structures.

References

1. B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. In *CCC*, pages 260–274, 2005.
2. O. Barkol, Y. Ishai, and E. Weinreb. On locally decodable codes, self-correctable codes, and t-private PIR. In *APPROX-RANDOM*, pages 311–325, 2007.
3. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *STOC*, pages 479–488, 1996.
4. D. Beaver. Precomputing oblivious transfer. In *Crypto*, pages 97–109, 1995.
5. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Crypto*, pages 62–76, 1991.
6. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *Journal of Cryptology*, 10(1):17–36, 1997.
7. A. Beimel, A. Gal, and M. Paterson. Lower bounds on monotone span programs. In *FOCS*, pages 674–681, 1995.
8. A. Beimel, O. Farras, and Y. Mintz. Secret sharing for very dense graphs. In *Crypto*, pages 144–161, 2012.
9. A. Beimel and Y. Ishai. On the power of nonlinear secret sharing. In *CCC*, pages 188–202, 2001.
10. A. Beimel, Y. Ishai, and E. Kushilevitz. General constructions for information-theoretic private information retrieval. *Jour. Comput. Syst.& Sci.*, 71(2):213–247, 2005.
11. A. Beimel, Y. Ishai, E. Kushilevitz, and I. Orlov. Share conversion and private information retrieval. In *CCC*, pages 258–268, 2012.
12. A. Beimel and T. Malkin. A quantitative approach to reductions in secure computation. In *TCC*, pages 238–257, 2004.
13. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computations. In *STOC*, pages 1–10, 1988.
14. R. Bendlin, I. Damgard, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *Eurocrypt*, pages 169–188, 2011.
15. 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.

16. 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.
17. G. Brassard, C. Crepeau, and J.-M. Robert. Information theoretic reduction among disclosure problems. In *FOCS*, pages 168–173, 1986.
18. S. Bublitz. Decomposition of graphs and monotone formula size of homogeneous functions. In *Acta Informatica*, pages 689–696, 1986.
19. D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *STOC*, pages 11–19, 1988.
20. X. Chen, N. Kayal, and A. Wigderson. Partial derivatives in arithmetic complexity and beyond. In *FSTTCS*, pages 1–138, 2011.
21. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS*, pages 41–50, 1995.
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. I. Damgard and S. Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC*, pages 621–641, 2013.
25. Y. Dodis and S. Micali. Parallel reducibility for information-theoretically secure computation. In *Crypto*, pages 74–92, 2000.
26. Z. Dvir, P. Gopalan, and S. Yekhanin. Matching vector codes. In *FOCS*, pages 705–714, 2010.
27. K. Efremenko. 3-query locally decodable codes of subexponential length. In *STOC*, pages 39–44, 2009.
28. P. Erdos and L. Pyber. Covering a graph by complete bipartite graphs. In *Discrete Mathematics*, pages 249–251, 1997.
29. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *Crypto*, pages 205–210, 1983.
30. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation (extended abstract). In *STOC*, pages 554–563, 1994.
31. O. Goldreich. Foundations of cryptography - volume 2, basic applications. 2004.
32. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
33. O. Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In *Crypto*, pages 73–86, 1988.
34. D. Harnik, Y. Ishai, and E. Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Crypto*, pages 284–302, 2007.
35. D. Harnik, Y. Ishai, E. Kushilevitz, and J. Nielsen. OT-combiners via secure computation. In *TCC*, pages 393–411, 2008.
36. Y. Ishai. Randomization techniques for secure computation. In *Secure Multi-Party Computation. Eds:* M. Prabhakaran and A. Sahai, 2013.
37. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Crypto*, pages 145–161, 2003.
38. Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, pages 174–184, 1997.
39. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.
40. Y. Ishai and E. Kushilevitz. On the hardness of information-theoretic multiparty computation. In *Eurocrypt*, pages 439–455, 2004.

41. Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, pages 600–620, 2013.
42. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Extracting correlations. In *FOCS*, pages 261–270, 2009.
43. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *Crypto*, pages 572–591, 2008.
44. M. Kaplan, I. Kerenidis, S. Laplante, and J. Roland. Non-local box complexity and secure function evaluation. In *FSTTCS*, pages 239–250, 2009.
45. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
46. J. Kilian. More general completeness theorems for secure two-party computation. In *STOC*, pages 316–324, 2000.
47. V. Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In *Asiacrypt*, pages 136–155, 2005.
48. E. Kushilevitz and N. Nisan. Communication complexity. 1997.
49. O. Lupanov. A method of circuit synthesis. In *Izvesitya VUZ, Radiofizika*, pages 120–140, 1958.
50. Y. Mintz. Information ratios of graph secret-sharing schemes. Master's thesis, Ben Gurion University, Israel, 2012.
51. V. Prabhakaran and M. Prabhakaran. Assisted common information with an application to secure two-party sampling. In *arxiv:1206.1282v1*, 2012.
52. M. O. Rabin. How to exchange secrets with oblivious transfer. In *Technical Report TR-81, Aiken Computation Lab, Harvard University*, 1981.
53. C. Shannon. The synthesis of two-terminal switching circuits. In *Bell System Technical Journal*, pages 59–98, 1949.
54. H. Sun and S. Shieh. Secret sharing in graph-based prohibited structures. In *INFOCOM*, pages 718–724, 1997.
55. M. van Dijk. On the information rate of perfect secret sharing schemes. In *Designs, Codes and Cryptography*, pages 143–169, 1995.
56. S. Winkler and J. Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In *Crypto*, pages 707–723, 2010.
57. D. Woodruff and S. Yekhanin. A geometric approach to information-theoretic private information retrieval. *SIAM J. Comp.*, 37(4):1046–1056, 2007.
58. S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *STOC*, pages 266–274, 2007.

A Multiparty Secure Computation

Notation. Let $\mathcal{F}_N^k$ denote the set of all boolean functions from $[N]^k$ to $\{0,1\}$. We will interpret $f \in \mathcal{F}_N^k$ as a k-party function. Also, we consider t-private k-server PIR for $\mathcal{F}_N^k$, a natural generalization of 1-private k-server PIR for $\mathcal{F}_N$ defined in Section 2.

The following theorems summarize the connections between t-private k-server PIR, and multiparty secure computation in the plain model, OT-hybrid model, and the preprocessing model. The protocols implied by the theorems are straightforward extensions of the ideas behind the protocols of Sections 4 and 5.

Theorem 5. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a t-private k-server PIR scheme for $\mathcal{F}_N^k$. Then, for any k-party functionality $f : [N]^k \to \{0,1\}$, the following hold:*

– *There is a perfectly secure k-party protocol π that realizes f in the* plain *model, tolerates $t < k/2$ passively corrupt parties, and has communication complexity $O(k^2 \cdot (\tau(\mathcal{Q}) + \tau(\mathcal{R})))$.*
– *There is a perfectly secure k-party protocol π that realizes f in the* OT-hybrid *model, tolerates $t < k$ passively corrupt parties, and has communication complexity $O(k^2 \cdot (\tau(\mathcal{Q}) + \tau(\mathcal{R})))$.*

Theorem 6. *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a t-private $(k+1)$-server PIR scheme for $\mathcal{F}_N^k$. Then, for any k-party functionality $f : [N]^k \to \{0,1\}$, there is a perfectly secure k-party protocol π that realizes f in the* preprocessing *model, and tolerates $t < k$ passively corrupt parties, and has correlated randomness complexity (and communication complexity) $O(k^2 \cdot (\tau(\mathcal{Q}) + \tau(\mathcal{R})))$.*

Plugging in parameters from the best known t-private k-server (resp. $(k+1)$-server) PIR protocols [10, 57] in Theorem 5 (resp. Theorem 6), we obtain the following corollary.

Corollary 4. *Let $f : [N]^k \to \{0,1\}$ be any k-party functionality. Then,*

– *There is a perfectly secure k-party protocol π that realizes f in the* plain *model, tolerates $t < k/2$ passively corrupt parties, and has communication complexity $N^{k/\lfloor 2k-1/t \rfloor} \cdot \text{poly}(k)$.*
– *There is a perfectly secure k-party protocol π that realizes f in the* OT-hybrid *model, tolerates $t < k$ passively corrupt parties, and has communication complexity $N^{k/\lfloor 2k-1/t \rfloor} \cdot \text{poly}(k)$.*
– *There is a perfectly secure k-party protocol π that realizes f in the* preprocessing *model, tolerates $t < k$ passively corrupt parties, and has correlated randomness complexity (and communication complexity) $N^{k/\lfloor 2k+1/t \rfloor} \cdot \text{poly}(k)$.*

We point out that for the specific case of $t = k-1$ our protocol in the OT-hybrid model has communication complexity $N^{k/2} \cdot \text{poly}(k)$ which improves over prior work which had complexity $N^{k-1} \cdot \text{poly}(k)$ [34].

For the case of honest majority, it is possible to obtain better results for t-private k-party computation when $k \geq 3^t$ via the best known t-private 3^t-server PIR protocols obtained by boosting (via [2]) the PIR protocols of [58, 27, 11].

Corollary 5. *For any $t \geq 0$, and for any $k \geq 3^t$-party functionality $f : [N]^k \to \{0,1\}$, there is a protocol π that realizes f in the plain model, and has the following features:*

– *π is perfectly secure, and tolerates t passively corrupt parties;*
– *The total communication complexity is $2^{\widetilde{O}(\sqrt{k \log N})} \cdot \text{poly}(k)$.*

B Extension to the Malicious Setting

In this section, we show how to compile our semihonest secure protocols for secure computation in the OT-hybrid/preprocessing/plain model in to malicious secure protocols for secure computation in the respective models. The high level idea is to use the IPS compiler [43], which is parameterized by an outer malicious secure protocol (that helps computing the target function) and an inner semi-honest secure protocol (for simulating the next message function of the outer protocol). The main challenge is in implementing the compiler while somewhat preserving the complexity of the underlying semihonest secure protocol.

To this end, the outer protocol that we employ is inspired by the instance hiding scheme of Beaver et al. [6]. If f represents the target function that we need to realize, then we set the target function of the outer protocol, say g to be, for parameter m, an m-variate degree-d polynomial over $\mathbb{F}$ obtained by *arithmetizing* f. To evaluate a function g, our outer protocol will use k parties (where k depends on the size of the input domain N), that evaluate g on *shares* of the actual input. Note that (1) the actual parties need to distribute shares computed from the joint input of both parties to the k virtual parties, and (2) each of the k virtual parties compute their next message which is the evaluation of g on the share they received. The share computation step depends only on the length of g's input, and the number of virtual parties. To evaluate g, the actual parties first interpret g as a boolean function g^* (with multi-bit output), and then use our semihonest secure protocol multiple times to evaluate each output bit of g^*. In other words, our semihonest secure protocol acts as the IPS compiler's inner protocol. The final output is obtained as in the scheme of [6] via polynomial interpolation, which in our compiled protocol will be performed using a secure computation protocol.

We summarize the discussion by stating the final theorems that we obtain, and defer the proofs to the full version.

Theorem 7. *Let σ be a statistical security parameter. For all $\epsilon > 0$, and for any 2-party functionality $f : [N] \times [N] \to \{0,1\}$, there is a protocol π that realizes f in the OT-hybrid model; this protocol is statistically secure against malicious parties, and has total communication complexity (including communication with the OT oracle) $\widetilde{O}(N^{\frac{2}{3}+\epsilon}) + \mathrm{poly}(\sigma, \log N, 1/\epsilon)$.*

Theorem 8. *Let σ be a statistical security parameter. 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 statistically secure against malicious parties, and has total communication complexity and correlated randomness complexity $2^{\widetilde{O}(\sqrt{\log N})} + \mathrm{poly}(\sigma, \log N)$.*

Theorem 9. *Let σ be a statistical security parameter. For any 3-party functionality $f : [N] \times [N] \times [N] \to \{0,1\}$, there is a protocol π that realizes f in the plain model; this protocol is statistically secure against a single malicious party, and has total communication complexity $2^{\widetilde{O}(\sqrt{\log N})} + \mathrm{poly}(\sigma, \log N)$.*