Perfect Secure Computation in Two Rounds *

Benny Applebaum† Zvika Brakerski‡ Rotem Tsabary‡

Abstract

We show that any multi-party functionality can be evaluated using a two-round protocol with perfect correctness and perfect semi-honest security, provided that the majority of parties are honest. This settles the round complexity of information-theoretic semi-honest MPC, resolving a longstanding open question (cf. Ishai and Kushilevitz, FOCS 2000). The protocol is efficient for NC^1 functionalities. Furthermore, given black-box access to a one-way function, the protocol can be made efficient for any polynomial functionality, at the cost of only guaranteeing computational security.

Our results are based on a new notion of *multi-party randomized encoding* which extends and relaxes the standard notion of randomized encoding of functions (Ishai and Kushilevitz, FOCS 2000). The property of a multi-party randomized encoding (MPRE) is that if the functionality g is an encoding of the functionality f, then for any (permitted) coalition of players, their respective outputs and inputs in g allow them to simulate their respective inputs and outputs in f, without learning anything else, including the other outputs of f. We further introduce a new notion of *effective algebraic degree*, and show that the round complexity of a functionality f is characterized by the degree of its MPRE. We construct degree-2 MPREs for general functionalities in several settings under different assumptions, and use these constructions to obtain two-round protocols. Our constructions also give rise to new protocols in the client-server model with optimal round complexity.

1 Introduction

Secure multi-party computation (MPC) is perhaps the most generic cryptographic task. A collection of n parties, each with its own input x_i, wish to jointly compute function of all of their inputs $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ so that each party learns its y_i and nothing else, and even a coalition of adversarial players should not learn more than the collection of outputs of its members. Throughout this work, we will be concerned with the most basic variant of this problem, which is often called *private computation*, where even adversarial parties are assumed to follow the protocol but try to learn as much as they can from their view (a.k.a semi-honest adversaries). Unless stated otherwise, we further assume that the adversary is computationally unbounded, and consequently require information-theoretic (perfect) privacy.

*This work is partially based on an extended abstract that appeared in the proceedings of TCC 2018 [ABT18].

†Tel-Aviv University, bennyap@post.tau.ac.il. Supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under grant agreement no. 639813 ERC-CLC, and the Check Point Institute for Information Security.

‡Weizmann Institute of Science, {zvika.brakerski,rotem.tsabary}@weizmann.ac.il. Supported by the Israel Science Foundation (Grant No. 468/14), Binational Science Foundation (Grants No. 2016726, 2014276), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

The seminal works of Ben-Or, Goldwasser and Wigderson [BGW88] and Chaum, Crépeau and Damgård [CCD88] established that in this setting security for non-trivial functions can only be achieved if the adversarial coalition includes strictly less than half of the total number of parties (a.k.a honest majority). They showed that in the presence of honest majority, any function f can be privately computed, thus existentially resolving the problem.

However, as with all computational tasks, one wishes to minimize the resources required to carry out an MPC protocols. A resource that received much attention is the round complexity: the number of rounds of communication required to carry out the protocol. We consider the standard simultaneous communication model where at each round each party can send a message to any other party, but these messages can only depend on information received in previous rounds. The aforementioned [CCD88, BGW88] solutions depend on the (multiplicative) depth of (the arithmetic representation of) the function f. For depth d, they require d rounds of communication (and the communication and computational complexity are polynomial in the number of parties n and the circuit size of f). In terms of lower bound, it is not hard to show that most functions cannot be privately computed with less than two rounds, but no better lower bound is known.

Constant-round information-theoretic protocols were first constructed by Bar-Ilan and Beaver [BB89] and were later extended in several works (cf. [FKN94]). Ishai and Kushilevitz [IK00, IK02] approached the 2-round lower bound: They presented a 3-round protocol, and in fact showed that a 2-round protocol is possible if instead of honest majority one requires that more than two-thirds of the parties are honest. Ishai and Kushilevitz note that their methods fall short of achieving the ultimate result and leave it as an open problem to resolve whether it is possible to achieve 2-round honest-majority protocol for all functions [IK00, Section 6]:

> *"An open question of a somewhat different flavor is that of finding the exact number of rounds required for privately evaluating an arbitrary (i.e., a worst-case) function f with an optimal privacy threshold. Using randomizing polynomials, an upper bound of 3 was obtained. If this bound is tight (i.e., 2 rounds are not enough) then, in a very crude sense, the randomizing polynomials approach is non-restrictive."*

In this work, we resolve this open question. We show that indeed any functionality can be privately computed in a 2-round protocol that only requires honest majority. The communication and computational complexity are asymptotically comparable to previous solutions.

Theorem 1.1 (2-round unconditional MPC)**.** *At the presence of honest majority, privately computing any functionality with perfect correctness and perfect privacy reduces non-interactively to the task of privately computing a degree-2 functionality. Consequently, in this setting, any function f can be privately computed in two rounds with polynomial efficiency in the number of parties and in the size of the formula (or even branching program) that computes f.*[1]

Furthermore, under the assumption that one-way functions exist, it is possible to improve the computational and communication complexity to polynomial in the size of the circuit computing f (rather than its formula size or exponential in the circuit depth), at the cost of only achieving computational security. Note that the honest majority condition cannot be lifted in this setting (unless one-way functions imply oblivious transfer).

[1] Branching programs (BP) are believed to be more powerful than formulas since the latter are restricted to computing NC^1 functions whereas the former can compute (possibly strictly) richer classes, including LOGSPACE and beyond.

Theorem 1.2 (2-round MPC in minicrypt)**.** *Assume the existence of one-way functions. Then, privately computing any polynomial-time functionality with computational privacy and honest majority reduces non-interactively to the task of privately computing a polynomial-time computable degree-2 functionality. Consequently, in this setting, any function f can be privately computed in two rounds with polynomial efficiency in the number of parties and the circuit size of f. The protocol makes only black-box use of the one-way function.*

Prior to this work, Beaver, Micali and Rogaway [BMR90, Rog91] (henceforth BMR) constructed the first constant-round computationally private MPC assuming honest majority and one-way functions. A careful analysis of their construction leads to 3 rounds.[2]

See Section 1.3 below for comparison with recent related results such as [ACGJ18, GIS18].

The Client-Server Setting. Our results extend to the so-called client-server setting [DI05], which considers a communication graph of the following form: A set of n *clients* that have inputs send messages (in a single round) to a set of m *servers*, the servers perform local computation and send messages (in a single round) back to the clients, who can then recover their outputs.

Ideally we would like to only require semi-honest security whenever we have honest majority among servers (and allow arbitrary corruption of clients). A straightforward extension of our methods from above achieves semi-honest security either information theoretically or computationally relying on one-way functions, but only so long as there is an honest majority among servers *and* and an honest majority among clients. We proceed to show how to achieve the stronger notion (with computational privacy) based on non-black-box use of oblivious transfer. (This point will be further discussed towards the end of Section 1.1.)

Theorem 1.3 (2-round MPC in the client-server model)**.** *Any n-party functionality f can be privately computed with two rounds in the client-server model with n clients and any number of m servers with each of the following guarantees:*

- *Perfect correctness and perfect privacy against any adversary that corrupts a minority of the clients and a minority of the servers. The complexity is polynomial in the formula size (or branching program size) of f and in n and m.*
- *Perfect correctness and computational privacy against any adversary that corrupts a minority of the clients and a minority of the servers. The protocol makes a black-box use of one-way functions and the complexity is polynomial in the circuit size of f and in n and m.*
- *Perfect correctness and computational privacy against any adversary that corrupts an* arbitrary *subset of the clients and a minority of the servers. The protocol makes a non-black-box use of (possibly multi-round) oblivious-transfer and the complexity is polynomial in the circuit size of f and in n and m.*

The concurrent work of [GIS18] shows that the last item can be strengthened to arbitrary subset of the clients and the servers at the expense of relying (in a non-black-box way) on the existence of a *two-round* oblivious transfer protocol. This result is therefore incomparable to the last item of Theorem 1.3.

[2] Throughout the paper, we refer to the simplified version of the BMR protocol that appears in Rogaway's thesis [Rog91].

1.1 Our Techniques

Ishai and Kushilevitz introduced the notion of randomizing polynomials, which was since generalized to the notion of randomized encoding (RE) [AIK04]. A function f is encoded by a function g if the output of g allows to reconstruct the output of f but nothing more. The [IK00] result essentially shows that any function f can be encoded by a function g of multiplicative degree 3 (over the binary field). Thus, instead of applying the [CCD88, BGW88] protocol to compute the function f directly, it is possible to apply it to compute g (the encoding of f). Since degree 3 functions can be computed in 3 rounds with honest majority, or in 2 rounds if more than two-thirds of the parties are honest, the encoding of [IK00, IK02] implies MPC protocols with these properties for all functions. We note that the computational complexity and output length of the encoding g may be significantly larger than those of f and indeed scale (roughly) polynomially with its formula size. An additional minor caveat is that the encoding g is a randomized function, even if f was deterministic. This is resolved using the standard technique of secret sharing the random tape between all users, i.e. each user holds private randomness and the function g is computed with a random tape that is the XOR of all private tapes. This transformation does not effect the multiplicative degree and therefore does not change the round complexity of the resulting protocol (though it incurs a poly(n) factor in computational and communication complexity).

It is evident from the above outline that if one could find a RE with multiplicative degree 2, the round complexity of MPC will be resolved. However, it was shown in [IK00] that such randomized encodings do not exist, at least if perfect correctness and security are sought (we recall that our solution achieves perfect correctness and security). The quotation above therefore suggests that the resolution of the round complexity of MPC will also resolve the question of optimality of the RE approach to the problem.

In this work, we show that indeed RE is too restrictive to resolve the round complexity problem. We present a natural generalization that we call multi-party randomized encoding (MPRE). This object allows to analyze randomized encodings in the specific context of MPC, and naturally translate it to protocols similarly to RE. While RE encodes a *computation* and ignores the partitioning of inputs between the parties, an MPRE takes into account the way that inputs and outputs are distributed among parties. Correspondingly, this notion of encoding allows to encode a *multiparty functionality* by another *multiparty functionality* (in contrast to the RE notion which allows to encode a function by another function). In this sense MPRE is much closer in spirit to MPC protocols, and one can easily go from protocols to MPREs and back. Being a *multiparty functionality*, in MPRE inputs are split between different parties who may also employ private local randomness (which does not make sense in the context of standard RE). The round complexity of the protocol induced by the MPRE depends on the *effective degree*, which allows preprocessing of local randomness. Theorem 1.1 follows by showing that any functionality has MPRE with effective degree 2 which is private against adversarial minority.

Multi-Party Randomized Encoding (MPRE). The definition of MPRE is inspired by that of RE, but with the emphasis that inputs and outputs can belong to different players. If we consider a multi-party functionality $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, then an MPRE of f would be a randomized functionality

$$g((x_1, r_1), \ldots, (x_n, r_n); s) = (z_1, \ldots, z_n),$$

where s is a global random string (which, we bear in mind, will be shared among users when a protocol is to be derived) and r_i is the local random string of player i. Decoding requires that

for each i, y_i can be recovered from z_i. The privacy requirement is that for any "legitimate" adversarial coalition $A \subseteq [n]$, the r and z values of all players in A can be simulated given their x and y values. In the context of honest majority we can consider protecting against all A of cardinality strictly less than $n/2$, but the MPRE notion is more general and allows some function classes to be encoded while allowing *any* adversarial $A \subseteq [n]$ (indeed we show such an encoding for a useful class). It is possible to show the expected composition theorem, arguing that if g is MPRE of f which is private against some class of adversarial coalitions $\mathcal{A}_1 \subseteq 2^{[n]}$, and there is a protocol that privately computes g against some class of adversarial coalitions $\mathcal{A}_2 \subseteq 2^{[n]}$, then the same protocol (augmented with local decoding) can be used to compute f, and is private against $\mathcal{A}_1 \cap \mathcal{A}_2$. It thus follows that if g is MPRE of f which is private against all adversarial minorities, and if g has effective degree 2 (allowing preprocessing of local randomness), then f has a 2-round protocol which is private against any adversarial minority.[3] Showing that all functions have such encoding will be our goal towards proving Theorem 1.1. For formal definitions of MPRE, composition and relation to other notions see Section 3.

How to Encode Any Function. As explained above, our goal is to show that any functionality $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ has an encoding that is both secure against all adversarial minorities and has effective degree 2. We do this in a sequence of steps. The first step is noticing that we can get a "friendly" MPRE from *any* protocol for computing f, even one with many rounds. We stress that this will not be our final MPRE. The definition of this MPRE g is straightforward: the output of party i is simply its view in the protocol, augmented with all the intermediate values computed locally by i. Note that this new functionality now requires local randomness of the parties. The fact that these views were generated by a protocol will be of particular use to us since the outputs of g can be viewed as wires of a boolean circuit, where each wire belongs to a different party in the computation. The view of each party in the protocol (i.e. its output in the functionality g) consists of values that it received from other parties, and values that it computed locally. We can thus envision a circuit whose gates are "owned" by players, and there are additional syntactic "transmission gates" that represent a message passing from one player to the other. Transmission gates do not have any functionality but rather represent change of ownership, still they will be useful for our next step. We call such MPREs "protocol compatible" and describe their properties formally in Section 4. Specifically, we will consider the MPRE induced (essentially) by the 3-round protocol that is based on a combination of [IK02] and [BGW88].

By employing a composition theorem for MPRE, it suffices to encode the functionality g by an MPRE h of effective degree 2. Indeed, we show that any *protocol-compatible functionality* g (i.e. one whose outputs can be represented as local views of parties in a multi-party protocol, or equivalently as wires of a circuit of the structure described above), can be encoded with effective degree-2 and privacy against *any* adversarial coalition. The MPRE takes great resemblance to the well known RE scheme that is based on information-theoretic garbled circuit [IK02]. (Specifically, it is based on the point-and-permute variant of Yao's garbled circuit [Yao86, BMR90, Rog91].) This randomized encoding scheme takes a circuit, and for each wire it samples two wire keys and a permutation bit, and its output is a list of "garbled tables" together with the permutation bits of the output wires. Expressing this in algebraic form leads to degree 3 randomized encoding. More generally, the degree of each garbled gate G is $\deg(G) + 1$.

[3] In fact, we show that the computation of f privately reduces to g via a non-interactive reduction that makes a single call to g.

In our MPRE, the wire keys will be sampled using the global randomness (which down the line is shared between all parties). Crucially, the permutation bits will be generated using the local randomness of the party that "owns" this wire, as per the protocol compatible functionality. One can verify that this description results in an encoding with effective degree 2. Indeed, the encoding consists of two type of gates: local-computation gates and transmission gates. In local-computation gates G, the input and output wires of the gate are owned by the same party, thus this party can preprocess the permutation bits and reduce the degree to 2. In the case of transmission gates, the fan-in is 1, and so the degree is only 2. The same proof as in [BMR90, Rog91, IK02, AIK06] can be used to show MPRE privacy. The construction is described in detail in Section 5.

Putting the two components together results in an MPRE h for every f which is secure against all adversarial minorities and has effective degree 2, giving rise to our final 2-round protocol. The computational and communication complexity are analyzed in the respective sections. Section 7 contains the proof of Theorem 1.1, putting together all relevant components.

The Computational Setting. Theorem 1.1 yields an inefficient protocol when applied to a general functionality which is computed by a polynomial-size circuit. Recall that the proof consists of two main steps: (1) encoding f by a protocol-compatible functionality g; and (2) encoding g by an MPRE with effective degree of 2. While the first step extends to arbitrary polynomial-size functionalities, the second part incurs a super-polynomial overhead due to the use of information-theoretic garbled circuits whose complexity is exponential in the depth of the encoded functionality.

We avoid this blow-up by replacing the information-theoretic garbled-circuit with a computational garbled circuit [Yao86, BMR90] (that can be viewed as a computational randomized encoding [AIK06]). Just like the information-theoretic setting, we modify this computational RE into a degree-2 MPRE by generating the permutation bit of each gate locally by the party who "owns" the gate. For more details see Section 6 (which is devoted to the construction of degree-2 computational MPRE) and Section 7 (which includes the proof of Theorem 1.2).

The Client-Server Setting and an Open Problem. MPREs are applicable to the client-server setting in an immediate manner. Let g be an MPRE of f which is secure against some class of adversarial coalitions $\mathcal{A}$ over n players. Assume that g can be computed in the client-server setting, with n clients and m servers, with security against a class $\mathcal{C}$ of client coalitions and a class $\mathcal{S}$ of server coalitions. Then f is computable in the client-server setting with security against $\mathcal{A} \cap \mathcal{C}$ client coalitions and $\mathcal{S}$ server coalitions.

In our setting, we show that all functions f have g with effective degree 2 and security against dishonest minority. One can verify that the protocols of [BGW88, CCD88], when applied to degree 2 functions, imply client-server protocols with security against arbitrary client collusion and dishonest server minority. The conclusion is that security is achieved if there is honest majority of both clients and servers. The first two items of Theorem 1.3 therefore follow immediately from the constructions of information-theoretic and computational degree-2 MPREs.

In order to remove the requirement for honest majority of clients in the client-server setting it suffices to construct a degree-2 MPRE security against an *arbitrary* collusion. Indeed, the third item of Theorem 1.3 is derived by providing such an MPRE based on a (non-black-box use) of an oblivious transfer protocol. Roughly speaking, the idea is to replace the first step of the MPRE construction with a protocol-compatible encoding g that is induced by a fully-secure protocol π for computing f. Since π employs oblivious-transfer, the MPRE g also depends on the code of

the oblivious transfer protocol. The second part of the reduction (encoding g via a computational MPRE based on garbled circuits) remains unchanged. (The construction of the MPRE appears in Theorem 7.3, and the full proof of Theorem 1.3 appears in Section 7.)

The question of constructing MPRE with effective degree-2 and security against arbitrary coalitions based on a black-box use of oblivious transfer or on a weaker assumption remains an interesting open problem.

1.2 Broader Perspective: Degree vs. Round Complexity

Since the pioneering constructions of perfect MPC [BGW88, CCD88], there appears to be a tight relation between the round complexity of privately computing a functionality f at the presence of honest majority to its algebraic degree. This relation was refined by [IK00], who showed that instead of considering the degree of f, one should focus on the degree of a RE $\hat{f}$ of f. Our work further replaces the notion of RE-degree with the *effective degree* of an MPRE $\hat{f}$ of f, and shows that this notion leads to an optimal round complexity. This finally proves that, when instantiated with the "right" notion of degree, the degree-based approach is indeed "non-restrictive" as suggested by Ishai and Kushilevitz.

It is instructive to take a closer look at the notion of effective degree and see how it relates to existing notions. Recall that effective degree essentially allows the parties to apply arbitrary local-preprocessing of their private randomness (and inputs) "for free", without charging it towards the degree. This relaxation is crucial for our results. Indeed, it can be shown that degree-d MPRE directly imply degree-d RE (see Remark 3.5). Also observe that the notion of effective degree inherently requires to treat the encoding $\hat{f}$ as a *multiparty functionality*, and therefore effective degree becomes meaningless in the case of RE. In this sense, MPRE is a convenient intermediate point between a protocol to RE; It takes into account the views of different players (which is crucial for defining effective degree) while being a non-interactive (and therefore easy to manipulate) object.

Let us further note that the methodology of degree-reduction via local preprocessing is not new. In particular, it is crucially employed in classical constant-round MPC protocols including Yao's two-party protocol [Yao86] and its multiparty variant [BMR90, Rog91]. Using our terminology, these protocols implicitly yield computational MPRE of constant effective degree. In particular, assuming one-way functions, Yao's protocol yields a computational MPRE of effective degree 2 for any efficiently computable 2-party functionality, and the BMR protocol yields a computational MPRE of effective degree 3 for any efficiently computable n-party functionality. Indeed, an important part of our conceptual contribution is to provide a formal, easy-to-handle, framework that captures this use of degree-reduction via preprocessing.

1.3 Other Related Works

Benhamouda and Lin [BL18] and Garg and Srinivasan [GS18] have recently constructed 2-round computationally-private protocols for arbitrary (efficiently computable) functions. This result is incomparable to Theorem 1.2: It does not require honest majority (i.e., privacy holds against arbitrary coalitions), but relies on a stronger computational assumption (the existence of two-message Oblivious Transfer, which is necessary in this setting). We further note that our high level approach shares some similarities with these works. Indeed, our notion of MPRE abstracts and generalizes the notion of *garbled protocols*, introduced by Garg and Srinivasan [GS17], which plays a key role in both [BL18] and [GS18].

Independently of our work, two recent papers study the notion of minimal round complexity for MPC in the honest majority setting. Ananth et al. [ACGJ18] focus on secure computation in the presence of certain types of active (malicious) adversaries, and present protocols under the assumption of honest majority in addition to some computational and/or setup assumptions. Most relevant to our work is a consequence of one of their result showing that based on one-way functions there is a 2-round protocol against semi-honest adversarial minority (in fact, they achieve a stronger notion called "security with abort"). Contrary to our work, the [ACGJ18] protocol is not applicable in the information theoretic setting, and therefore does not have bearing on the question of MPC with perfect security. Furthermore, our approach shows a reduction from the computation of general functionalities to the computation of degree-2 functionalities, which is not achieved by [ACGJ18] (even implicitly, as far as we can tell).

Garg, Ishai and Srinivasan [GIS18] study the construction of information theoretic security for semi-honest MPC in various settings. Most relevant to this work is their construction of a 2-round protocol with perfect security for formulas. However, in their protocol, unlike ours, communication complexity grows super-polynomially with the number of players. One can again attribute this to falling short of reducing the general MPC task to the task of computing degree-2 functionalities.

Subsequent work. The techniques of this paper were extended in [ABT19] to the malicious setting (i.e., to the case of active adversaries). This shows that degree-2 functionalities are "complete" under non-interactive reductions for maliciously-secure MPC. Consequently, [ABT19] derive new round-efficient (and in some cases round-optimal) maliciously-secure protocols for various settings (e.g., perfect, statistical and computational depending on the privacy threshold).

Acknowledgements. We are grateful to Yuval Ishai, Akshayaram Srinivasan, Muthuramakrishnan Venkitasubramaniam, and Hoteck Wee for valuable discussions and to the anonymous referees of TCC 2018 for carefully reading this paper and for providing us with helpful feedback.

1.4 Paper Organization

We begin with some general background on multiparty functionalities and secure multiparty computation in Section 2. In Section 3, we introduce the notion of multipatry randomized encoding, and discuss its properties. In Section 4 we show how to use MPC protocols (in particular [BGW88]) to obtain "protocol-compatible" MPRE, and in Section 5 show how to transform such an encoding into a degree-2 MPRE based on information-theoretic garbled circuits. The latter transformation suffers from an exponential dependency in the depth of the encoded function which is removed in Section 6 by relying on computationally-private garbled circuits. Section 7 uses these tools to prove our main theorems.

2 Preliminaries

This section defines multiparty functionalities and provides some basic background on secure computation. It will convenient to use a somewhat non-standard notation for functionalities, and so even an expert reader may want to read this part carefully. (In contrast, the MPC subsection can be safely skipped.)

2.1 Multi-Party Functionalities

An n-party functionality is a function that maps the inputs of n parties to a vector of outputs that are distributed among the parties. Without loss of generality, we assume that the inputs of each party are taken from some fixed input domain X (e.g., bit strings of fixed length). It will be convenient to represent a functionality by a pair $f : X^n \to \{0,1\}^m$ and $P : [m] \to 2^{[n]}$. The function f maps the joint inputs of all parties $x = (x_1, \ldots, x_n)$ to an output vector $y = (y_1, \ldots, y_m)$, and the mapping $P : [m] \to 2^{[n]}$ determines the distribution of outputs between the parties, i.e., the i-th output y_i should be delivered to the parties in the set $P(i)$. By default (and without loss of generality), we assume that $P(i)$ is always a singleton and therefore think of P as a mapping from $[m]$ to $[n]$. Sometimes the output partition function P will be implicit, and refer to f as a functionality. We further use the convention that, for a string $y = f(x)$ and a subset of parties $T \subseteq [n]$, the restriction of y to the coordinates held by the parties is denoted by $y[T] = (y_j)_{j:P(j)\in T}$. When $T = \{i\}$ is a singleton, we simply write $y[i]$.

We will also make use of *randomized functionalities*. In this case, we let f take an additional random input r_0 and view r_0 as an internal source of randomness that does not belong to any party. We typically write $f(x_1, \ldots, x_n; r_0)$ and use semicolon to separate the inputs of the parties from the internal randomness of the functionality.

Finally, a central notion in this work is that of *effective degree* of a functionality, which generalizes the standard notion of degree. A multi-output functionality f has degree D if each of its outputs can be written as an $\mathbb{F}_2$-polynomial of degree D over the deterministic and random inputs. Intuitively, the effective degree is the degree of the functionality if the parties are allowed arbitrary local preprocessing. A formal definition follows.

Definition 2.1 (Effective degree)**.** *A (possibly randomized) n-party functionality $f : X^n \times R' \to \{0,1\}^m$ has* effective degree *d if there exists a tuple of local preprocessing functions $(h_1, \ldots, h_n)$ and a degree-d function h such that*

$$h(h_1(x_1), \ldots, h_n(x_n); r') = f(x_1, \ldots, x_n; r'), \tag{1}$$

for every $x_1, \ldots, x_n$ and internal randomness r'.

2.2 Standard Background on Secure Computation

Through the paper, we assume a fully-connected network with point-to-point private channels. We focus on semi-honest (aka passive) secure computation hereafter referred to as *private* computation. (See, e.g., [Can00, Can01, Gol04], for more detailed and concrete definitions.)

Definition 2.2. (Private computation) *Let $f(x_1, \ldots, x_n)$ be a (possibly randomized) n-party functionality. Let π be an n-party protocol. We say that the protocol* τ-privately computes *f* with perfect privacy *if there exists an efficient randomized simulator* Sim *for which the following holds. For any subset of corrupted parties $T \subseteq [n]$ of size at most τ, and every tuple of inputs $x = (x_1, \ldots, x_n)$ the joint distribution of the simulated view of the corrupted parties together with output of the honest parties in an ideal implementation of f,*

$$\mathsf{Sim}(T, x[T], y[T]), \quad y[\bar{T}], \qquad \text{where } y = f(x) \text{ and } \bar{T} = [n] \setminus T,$$

is identically distributed to

$$\mathsf{View}_{\pi,T}(x), \quad \mathsf{Output}_{\pi,\bar{T}}(x),$$

where $\mathsf{View}_{\pi,T}(x)$ *and* $\mathsf{Output}_{\pi,\bar{T}}(x)$ *are defined by executing* π *on* x *with fresh randomness and concatenating the joint view of the parties in* T *(i.e., their inputs, their random coin tosses, and all the incoming messages), with the output that the protocol delivers to the honest parties in* $\bar{T}$*. The computational variant of the definition is obtained by settling for computational indistinguishability with respect to non-uniform polynomial-time adversaries.*

Secure Reductions. To define secure reductions, consider the following *hybrid* model. An n-party protocol augmented with an oracle to the n-party functionality g is a standard protocol in which the parties are allowed to invoke g, i.e., a trusted party to which they can securely send inputs and receive the corresponding outputs. The notion of τ-security generalizes to protocols augmented with an oracle in the natural way.

Definition 2.3. *Let* f *and* g *be* n*-party functionalities. A* τ perfectly-private *reduction from* f *to* g *is an* n*-party protocol that given an oracle access to the functionality* g*,* τ*-privately realizes the functionality* f *with perfect security. We say that the reduction is* non-interactive *if it involves a single call to* f *(and possibly local computations on inputs and outputs), but no further communication. The notions of* τ computationally-private *reduction is defined analogously.*

Appropriate composition theorems, e.g. [Gol04, Thms. 7.3.3, 7.4.3] and [Can00], guarantee that the call to g can be replaced by any protocol that τ-privately realize g, without violating the security of the high-level protocol for f.

3 Multi-Party Randomized Encodings

In this section we formally present the notion of *multi-party randomized encodings* (Section 3.2), relate it to MPC protocols (Section 3.4), and study its properties (Section 3.5). As discussed in the introduction, this new notion can be viewed as a relaxation of the more standard notion of randomized encoding of functions. (See Section 3.1).

3.1 Randomized Encoding of Functions

We begin with the standard notion of randomized encoding (RE) [IK00, AIK04]. In the following let X, Y, Z, and R be finite sets.

Definition 3.1 (Randomized Encoding [AIK04, AIK06])**.** *Let* $f : X \to Y$ *be a function. We say that a function* $\hat{f} : X \times R \to Z$ *is a* δ*-correct,* (t, ϵ)*-private* randomized encoding *of* f *if the following hold:*

- δ***-Correctness:*** *There exist a deterministic decoder* Dec *such that for any input* $x \in X$*,*

$$\Pr_{r \xleftarrow{\$} R}[\mathsf{Dec}(\hat{f}(x; r)) \neq f(x)] \leq \delta.$$

- (t, ϵ)***-Privacy:*** *There exists a randomized simulator* Sim *such that for any* $x \in X$ *and any circuit* Adv *of size* t

$$\left| \Pr[\mathsf{Adv}(\mathsf{Sim}(f(x))) = 1] - \Pr_{r \xleftarrow{\$} R}[\mathsf{Adv}(\hat{f}(x; r)) = 1] \right| \leq \epsilon.$$

We refer to the second input of $\hat{f}$ as its random input*, and a use semicolon (;) to separate deterministic inputs from random inputs.*

An encoding $\hat{f}$ is useful if it is simpler in some sense than the original function f. In the context of MPC the main notion of simplicity is the degree of the encoding, where the each output of $\hat{f}$ is viewed as a polynomial over (x, r). Other notions of simplicity have been used in other contexts. (See [Ish13, App17] for surveys on REs.)

3.2 MPRE Definition

Inspired by the notion of randomized encoding of functions [IK00, AIK04], we define the notion of multiparty randomized encoding (MPRE). Syntactically, we encode a *functionality* $f(x_1, \ldots, x_n)$ by a randomized functionality

$$\hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0)$$

that employs internal randomness $r_0 \in R$ and augments the input of each party by an additional random input $r_i \in R$, for some fixed domain R (by default bit-string of fixed length). Roughly speaking, the view of the encoding $\hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0)$ that is available to a subset T of parties (i.e., the parties inputs, randomness and outputs) should contain the same information that is revealed to the subset T by the functionality $f(x)$ (i.e., the inputs and outputs).

The following heavily relies on our (somewhat non-standard) formalization of multi-party functionalities, see Section 2.1.

Definition 3.2 (Multi-Party Randomized Encoding (MPRE))**.** *Let $f : X^n \to \{0,1\}^m$ be an n-party deterministic functionality with an output partition $P : [m] \to [n]$. We say that an n-party randomized functionality $\hat{f} : (X \times R)^n \times R \to \{0,1\}^s$ with output partition Q is a multi-party randomized encoding of f with privacy threshold of τ if the following hold:*

- ***Perfect Correctness:*** *There exists a deterministic decoder* Dec *such that for every party $i \in [n]$, and every tuple of input-randomness pairs*

  $$((x_1, r_1), \ldots, (x_n, r_n)) \in (X \times R)^n$$

  and every internal randomness $r_0 \in R$ it holds that

  $$\mathsf{Dec}\,(i, \hat{y}[i], x_i, r_i) = y[i],$$

  where $y = f(x_1, \ldots, x_n)$, $\hat{y} = \hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0)$, and, recall that $\hat{y}[i]$ is the restriction of $\hat{y}$ to the coordinates delivered to party i by $(\hat{f}, Q)$, and $y[i]$ is the restriction of y to the coordinates delivered to party i by (f, P).[4]

- *(τ, t, ϵ)-**Privacy:*** *There exists a randomized simulator* Sim *such that for every set $T \subseteq [n]$ of parties of size at most τ and every set of inputs $x = (x_1, \ldots, x_n)$ it holds that the random variable*

  $$\mathsf{Sim}(T, x[T], y[T]), \qquad \text{where } y = f(x_1, \ldots, x_n)$$

  and the random variable

  $$(x[T], r[T], \hat{y}[T]),$$

[4] As in the case of RE, one can relax correctness and allow a small decoding error. Since all our constructions natively achieve perfect correctness, we do not define this variant.

where

$$\hat{y} = \hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0), \text{ and } (r_0, r_1, \ldots, r_n) \xleftarrow{\$} R^{n+1},$$

cannot be distinguished by a t-size circuit with advantage better than ϵ.

We say that privacy is perfect *if (τ, t, ϵ)-privacy holds for any t and $\epsilon = 0$. We always represent an MPRE $\hat{f}$ by a Boolean circuit that computes $\hat{f}$, and define the* size *and* depth *of $\hat{f}$ to be the size and depth of the corresponding circuit. We refer to the randomness r_0 as the* internal randomness *of the encoding. When such randomness is not used, we refer to $\hat{f}$ as an MPRE with no internal randomness.*

Observe that any functionality trivially encodes itself. Indeed, MPRE $\hat{f}$ becomes useful only if it is simpler in some sense than f. Jumping ahead, our main notion of simplicity will be effective degree.

Remark 3.1 (Perfect and Computational encodings of infinite functionalities)**.** *Definition 3.2 naturally extends to an infinite sequence of functionalities $f = \{f_\lambda\}_{\lambda \in \mathbb{N}}$ where f_λ is an $n(\lambda)$-party functionality whose domain, range, and complexity may grow polynomially with λ. We say that a sequence of $n(\lambda)$-party functionalities $\hat{f} = \{\hat{f}_\lambda\}_{\lambda \in \mathbb{N}}$ is a perfectly correct $(\tau(\lambda), t(\lambda), \epsilon(\lambda))$-private MPRE of f if there exists an efficient algorithm (compiler) which gets as an input 1^λ and outputs (in time polynomial in λ) three circuits $(\hat{f}_\lambda, \mathsf{Dec}_\lambda, \mathsf{Sim}_\lambda)$ which form a perfectly correct $(\tau(\lambda), t(\lambda), \epsilon(\lambda))$-private MPRE of f_λ. We refer to an MPRE as* perfect *(abbreviated pMPRE) if the above holds for any function $t(\cdot)$ and for $\epsilon = 0$, and refer to it as being* computational *(abbreviated cMPRE) if the above holds for $t(\lambda) = \lambda^{\omega(1)}$ and $\epsilon(\lambda) = 1/\lambda^{\omega(1)}$. Similar extensions applies to REs (as was done in previous works). Throughout this work, we mainly ignore uniformity issues and treat all functionalities as finite. However, all our MPRE constructions satisfy strong uniformity properties. That is, the encoder, decoder, and simulator can be all constructed efficiently given the description of the encoded functionality (say as a circuit). Hence, the encoding inherits the uniformity properties of the encoded functionality.*

Remark 3.2. *The parameter λ is being used to quantify both the complexity of f (circuit size and input length) and the security level (computational privacy). When describing some of our constructions, it will be convenient to separate between these two different roles and treat λ solely as a security parameter (independently from the complexity of f). Computational privacy will be guaranteed (in the sense of the above definition) as long as λ is set to be polynomial in the complexity of f.*

3.3 MPRE vs. RE

Remark 3.3 (MPRE: equivalent formulation)**.** *Let $f : X^n \to \{0,1\}^m$ be an n-party deterministic functionality let $\hat{f} : (X \times R)^n \times R \to \{0,1\}^s$ be an n-party randomized functionality. For a coalition T, consider the T-restricted function $f_T(x)$ that computes $y = f(x)$ and outputs $(x[T], y[T])$, i.e., the input and output information that belongs to the coalition T. Similarly, let $\hat{f}_T(x; (r_i)_{i\in[n]}, r_0)$ denote the randomized function that computes $\hat{y} = \hat{f}((x_i, r_i)_{i\in[n]}; r_0)$ and outputs $(x[T], r[T], \hat{y}[T])$, i.e., all the information that belongs to the coalition T. It is not hard to show that $\hat{f}$ is an MPRE of f with perfect correctness and (τ, t, ϵ)-privacy if and only if for every coalition T of size at most τ, the randomized function $\hat{f}_T$ is a perfectly-correct (t, ϵ)-private randomized encoding of the function f_T. Indeed, the privacy property of the MPRE for a coalition T is identical to the privacy*

property for $\hat{f}_T$. Similarly, for a singleton T, the correctness property of the MPRE is identical to the correctness property for the RE $\hat{f}_T$. Finally, for every lager coalition T, the correctness of $\hat{f}_T$ follows from the correctness over singletons.

In order to carry this equivalence to the case of uniform MPRE for a collection of functions (as in Remark 3.1) one has to require a uniformity property over the encodings $\hat{f}_T$. Specifically, we require the existence of an efficient compiler that gets as an input a security parameter 1^λ and outputs (in time polynomial in λ) three circuits $(\hat{f}_\lambda, \mathsf{Dec}_\lambda, \mathsf{Sim}_\lambda)$ such that for every τ-bounded T, the T-restricted MPRE $\hat{f}_{\lambda,T}$ forms a perfectly-correct $(\tau(\lambda), t(\lambda), \epsilon(\lambda))$-private RE of the T-restricted function $f_{\lambda,T}$ with decoder $\mathsf{Dec}_\lambda(T,\cdot)$ and simulator $\mathsf{Sim}_\lambda(T,\cdot)$.

Remark 3.4 (RE as a special case of MPRE)**.** *The notion of RE forms a strong special case of MPRE. Consider an n-party functionality $F(x_1,\ldots,x_n)$ in which all the parties receive the same output denoted by $f(x_1,\ldots,x_n)$. Suppose that we have an MPRE $\hat{F}$ for F with the following special properties: (1) $\hat{F}$ employs only internal randomness r_0 and the parties do not use private randomness at all; (2) All parties get from $\hat{F}$ the same output, denoted by $\hat{f}(x_1,\ldots,x_n;r_0)$; (3) The decoder of each party i recovers $F(x_1,\ldots,x_n)$ solely based on $\hat{f}(x_1,\ldots,x_n;r_0)$ without using the input x_i. Then, the MPRE $\hat{F}$ induces an RE $\hat{f}$ for the function f (underlying the functionality F). In fact, the converse direction also holds and any RE $\hat{f}$ can be used to derive an MPRE with the above syntactic properties. (This leads to an alternative definition of REs). Observe that properties 1–3 guarantee two important semantic properties: (a) the MPRE achieves a maximal privacy threshold of n; and (b) such an MPRE $\hat{F}$ is insensitive to the way that the input $x = (x_1,\ldots,x_n)$ is distributed between the parties.*

Remark 3.5 (Extracting RE from MPRE)**.** *Given a function $f : X \to Y$, we can always extract an RE $\hat{f}$ from an MPRE $\hat{F}$ for a related functionality F. Fix an arbitrary integer $n \geq 2$, and define an n-party functionality F in which the input $x \in X$ is arbitrarily partitioned between the first $n-1$ parties, and the output to all parties (including the last one) is $f(x)$. Consider an MPRE $\hat{F}$ for F with privacy threshold 1, and let $\hat{F}_i$ denote the output that is delivered to the i-th party. Then, the function $\hat{f}(x;(r_0,\ldots,r_n)) := (\hat{F}_n(x;r_0,\ldots,r_n), r_n)$ is an RE of f. (That is, $\hat{f}$ maps the collective input x, the randomness r_i of each party, and the internal randomness r_0, to the output of the last player in $\hat{F}$ together with its randomness.) Indeed, MPRE correctness guarantees that $f(x)$ can be decoded from $\hat{f}(x;r)$ and MPRE privacy guarantees that the distribution $\hat{f}(x;r)$ can be simulated given $f(x)$. One interesting conclusion of the above is that, in terms of standard degree, MPRE is not more expressive than RE. In particular, if every finite functionality has a degree-d perfect MPRE then any finite function has a degree-d perfect RE. (The reverse implication also holds). Hence, in order to benefit from the more expressive power of MPRE one has to resort to other notions of degree (like effective degree).*[5]

3.4 From MPRE to MPC Protocol

The main motivation for studying MPRE's is the following simple observation.

Proposition 3.1. *Let f be an n-party functionality. Let g be a perfect (resp., computational) MPRE of f with privacy threshold of τ. Then, the task of τ-privately computing f with perfect privacy (resp., computational privacy) reduces non-interactively to the task of τ-privately computing g with perfect privacy (resp., computational privacy).*

[5] We thank Akshayaram Srinivasan for a discussion that lead to this point.

In particular, by using standard composition theorems any protocol π that τ-privately computes g with perfect (resp., computational) privacy can be turned into a protocol π' with the same complexity and round complexity that τ-privately computes f with perfect (resp., computational) privacy.

Proof. Let $f : X^n \to \{0,1\}^m$ be an n-party functionality. Let $g : (X \times R)^n \times R \to \{0,1\}^s$ be a perfect (resp., computational) MPRE of f with privacy threshold of τ, and let Dec_g and Sim_g be the decoder and simulator for g. We describe a τ-private non-interactive protocol π for f in which the parties make a single call to a g-oracle (with no further interaction). Recall that g may have internal randomness, and so it is viewed as an n-party randomized functionality that takes from each party an input (x_i, r_i), samples r_0, computes $\hat{y} = g((x_i, r_i)_{i\in[n]}, r_0)$ and delivers $\hat{y}[i]$ to the i-th party.

The protocol π proceeds as follows. At the beginning, the i-th party, who holds an input x_i, samples local randomness $r_i \xleftarrow{\$} R$. Then, the parties jointly call the functionality g with the private inputs $((x_1, r_1), \ldots, (x_n, r_n))$, and each party $i \in [n]$ receives the value $\hat{y}[i]$ as a result. At the end, the i-th party outputs the value $\mathsf{Dec}_g(i, x_i, r_i, \hat{y}[i])$.

We prove that π privately realizes f. Since f is a deterministic functionality, it suffices to establish correctness and privacy separately (cf. [Gol04, Section 7.2.2]). We begin with correctness. Consider an execution of π over some arbitrary vector of inputs $(x_1, \ldots, x_n) \in X^n$. By the correctness of g, it holds that, for every choice of internal randomness r_0, the value computed by the i-th party $\mathsf{Dec}_g(x_i, r_i, \hat{y}[i])$ where $\hat{y} = g((x_1, r_1), \ldots, (x_n, r_n), r_0)$, equals to the i-th output of $f(x_1, \ldots, x_n)$.

We move on to prove that the protocol is private. Fix some adversarial set $T \subseteq [n]$ of size at most τ. We use the MPRE simulator $\mathsf{Sim}_g(T, x[T], y[T])$ to map an input/output pair $(x[T], y[T])$ of the coalition T to its view $(x_T, r_T, \hat{y}[T])$ in π. The perfect privacy (resp., computational privacy) of the MPRE guarantees that for every fixing of $x = (x_1, \ldots, x_n)$, the simulated view $\mathsf{Sim}_g(T, x[T], y[T])$ where $y = f(x)$, is perfectly indistinguishable (resp., computationally indistinguishable) from the actual view of T in π (induced by the randomness of the honest parties and the internal randomness of g). □

3.5 Manipulating MPRE

One can always get rid of the internal randomness r_0 of an MPRE

$$\hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0)$$

by extending the randomness of each party with an additional random string r'_i and applying the functionality $\hat{f}$ with r_0 set to $\sum_i r'_i$. Here, we assume that the randomness domain R is a set of fixed length strings and so addition stands for bit-wise XOR. (More generally, this transformation works as long as "addition" forms a group operation over the randomness space R.) Formally, the following holds.

Proposition 3.2 (Removing internal randomness)**.** *Suppose that the functionality $\hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0)$ is a perfectly correct (τ, t, ϵ)-private MPRE of (f, P). Then the functionality*

$$g((x_1, r_1, r'_1), \ldots, (x_n, r_n, r'_n)) := \hat{f}((x_1, r_1), \ldots, (x_n, r_n); \sum_i r'_i)$$

is a perfectly correct (τ, t, ϵ)-private MPRE of (f, P).

Note that g has the same algebraic degree and the same effective degree as $\hat{f}$ over $\mathbb{F}_2$. (A multi-output functionality f has degree D if each of its outputs can be written as an $\mathbb{F}_2$-polynomial of degree D over the deterministic and random inputs. For effective degree see Definition 2.1.)

Composition (Re-Encoding MPRE). The composition property of REs ([AIK04, AIK06]) asserts that if we take an encoding $g(x;r)$ of $f(x)$, view it as a deterministic function $g'(x,r)$ over x and r, and re-encode this function by a another RE $h(x,r;r')$, then the function $h'(x;(r,r'))$ is an encoding of f. We prove a similar statement regarding MPRE's.

Lemma 3.3 (Composition)**.** *Let $(f(x_1,\ldots,x_n),P)$ be an n-party functionality and assume that the functionality $(g((x_1,r_1)\ldots,(x_n,r_n)),Q)$ perfectly encodes f with threshold τ_1 and no internal randomness. Further assume that the functionality $(h(((x_1,r_1),r'_1)\ldots,((x_n,r_n),r'_n);r'_0),M)$ perfectly encodes the functionality (g,Q) (viewed as a deterministic functionality over the domain $(X')^n$ where $X'=(X\times R)$ with threshold τ_2). Then, the functionality (h',M), where*

$$h'((x_1,(r_1,r'_1)),\ldots,(x_n,(r_n,r'_n));r'_0) := h(((x_1,r_1),r'_1)\ldots,((x_n,r_n),r'_n)),$$

is a perfect MPRE of f with threshold $\min(\tau_1,\tau_2)$.

(Observe that h' is defined identically to h except each party i treats x_i as its deterministic input of i and (r_i,r'_i) as its randomness.)

Proof. For inputs $x_1,\ldots,x_n$ and random strings $r_1,\ldots,r_n$ and $r'_1,\ldots,r'_n$, we let $y=f(x_1,\ldots,x_n)$, $z=g((x_1,r_1)\ldots,(x_n,r_n))$ and $w=h(((x_1,r_1),r'_1)\ldots,((x_n,r_n),r'_n))$. For any party i, we denote by $y[i]$ (resp., $z[i],w[i]$) the restriction of the string to the indices which are delivered to the i-th party by the corresponding functionality.

Correctness: Fix some party i. We decode $y[i]$ from $((i,x_i,(r_i,r'_i)),w[i])$ in two steps: (1) Call the h-decoder on $((i,(x_i,r_i),r'_i),w[i])$ to obtain $z[i]$; and (2) Call the g-decoder on $((i,r_i,x_i),z[i])$ to obtain $y[i]$. The perfect correctness of the two decoders implies that, for any $\vec{x},\vec{r},\vec{r'}$, the new decoder decodes properly.

Privacy: Fix some coalition T of size at most $\min(\tau_1,\tau_2)$. Given $x[T]$ and $y[T]$, we use the g-simulator to sample the random variables $(x[T],r[T],z[T])$ and feed them to the h-simulator that generates the random variables $(x[T],r[T],r'[T],w[T])$. By the perfect privacy of the simulators, for every input x, the simulated random variable is identically to the real MPRE distribution $(x[T],r[T],r'[T],w[T])$ induced by a uniform choice of r and r'. □

A similar lemma holds in the computational setting as well.

Lemma 3.4 (Composition (Computational version))**.** *Let $f=\{f_\lambda\}$ be an infinite family of $n(\lambda)$-party functionalities which is computationally encoded by the families of functionalities $g=\{g_\lambda\}$ with privacy threshold $\tau(\lambda)$ and with no internal randomness. Suppose that $h=\{h_\lambda\}$ computationally encode g with privacy threshold of $\tau'(\lambda)$. Then, (h',P), defined as in Lemma 3.3, forms a computational encoding of f with privacy threshold of $\min(\tau,\tau')$.*

4 Encoding via Protocol-Compatible Functionalities

In this section we show that any functionality f can be encoded by a so-called *protocol compatible* functionality g that enjoys "nice" syntactic properties.

4.1 From MPC Protocol to MPRE

We begin by noting that any protocol naturally induces an MPRE as shown below.

Definition 4.1 (The view functionality)**.** *Let π be an n-party protocol in which the i-th party holds a deterministic input x_i and private randomness r_i. The n-party* view functionality *g_π is defined as follows:*

- *The input of the i-th party is (x_i, r_i).*
- *The output of the i-th party consists of all the messages that are sent to her in an execution of π (on the inputs $(x_1, r_1), \ldots, (x_n, r_n)$).*

We also consider the extended view functionality *in which, in addition to the above, g_π delivers to each party i all intermediate values that are computed locally by i, where the local computation of every party is viewed as a Boolean circuit.*

Note that the view and extended view can deterministically be derived from each other.

Proposition 4.1. *Let π be a protocol that implements the n-party functionality $f(x_1, \ldots, x_n)$ with perfect correctness and perfect (resp., computational) privacy against a passive adversary that may corrupt up to τ players. Then the view functionality and the extended view functionality of π encode the functionality f with perfect correctness and perfect (resp., computational) privacy threshold τ.*

Proof. The proposition follows immediately from the fact that π privately implements f as per Definition 2.2. The correctness of π translates into correctness of g_π and the τ-privacy of the protocol immediately translates into τ-privacy of the MPRE. □

An extended view functionality g_π has several useful syntactic properties. These are captured by the following notion of *protocol compatible* functionality.

Definition 4.2. *A* protocol compatible *functionality (f, P) is a functionality with no internal randomness that can be represented by a Boolean circuit C as follows.*

- *The circuit C takes the same inputs as f. The outputs $y = (y_1, \ldots, y_m)$ of $f(x)$ consist of the values of all the wires in the circuit (including internal wires and input wires) sorted under some topological order (inputs are first).*
- *The computation in C is performed via two types of gates.*
  - *A* transmission gate *delivers a value from one party to another, i.e. it maps a single input y_a to a single output y_b such that $y_a = y_b$ and possibly $P(a) \neq P(b)$.*
  - *A* local computation gate *(wlog, NAND gate) maps two inputs (y_a, y_b) to a single output y_c, where $P(a) = P(b) = P(c)$.*

Proposition 4.2. *Let π be an n-party protocol and let g_π be its extended view functionality. Then, g_π is protocol compatible.*

Proof. By definition, every output bit of g_π is either an input bit, the result of some local computation, or some incoming message. □

Remark 4.1 (Extended view in a hybrid model)**.** *Consider a protocol π operates in a h-hybrid model where h is some n-party functionality. (Recall that this means that the parties can invoke a call to an ideal version of h.) In this case, the view functionality and the extended view functionality (which are still well defined) still form an MPRE of f just like in Proposition 4.1. However it will not satisfy the syntax of Definition 4.2.*

4.2 BGW-based MPRE

The extended view functionality of the semi-honest protocol from [BGW88], henceforth denoted BGW, gives rise to the following MPRE.

Theorem 4.3 (BGW-based protocol-compatible encoding)**.** *Every n-party functionality f can be perfectly encoded with threshold privacy of $\tau = \lfloor \frac{n-1}{2} \rfloor$ by a protocol-compatible MPRE g of size $O(S \cdot \mathrm{poly}(n))$ and depth $O(D \cdot \log n)$ where S denotes the circuit size of f and D denotes the multiplicative depth of f.*

Jumping ahead, we mention that in order to derive our main theorem with complexity which grows polynomially in the number of parties, it is crucial to make sure that the depth of g is at most logarithmic in n.

Proof. We consider the BGW protocol π for computing f against a passive adversary that corrupts up to τ parties. By Propositions 4.1 and 4.2, it suffices to show that π can be implemented so that its extended-view functionality g_π is of size $O(S \cdot \mathrm{poly}(n))$ and depth $O(D \cdot \log n)$.

Recall that π interprets f as an arithmetic circuit over a sufficiently large field $\mathbb{F}$ of size $|\mathbb{F}| > n$ and that each party i is associated with a fixed public field element $\alpha_i \in \mathbb{F}$ (as a property of the protocol and independently of the input). Thus the first n powers of each α_i are to be treated as pre-computed constants. The local computation L of every party for each multiplication gate (and for the input gates) can be implemented by a $\mathrm{poly}(n)$-size arithmetic circuit of constant depth whose addition gates have unbounded fan-in and the multiplication gates have fan-in 2. (Indeed, all local computation can be written as matrix-vector multiplications.) This gives rise to an arithmetic circuit with bounded fan-in gates, $\mathrm{poly}(n)$ size, $O(\log n)$ depth and *constant* multiplicative depth.

We continue by showing that such an arithmetic circuit L can be realized by a Boolean NC^1 circuit (of size $\mathrm{poly}(n)$, depth $O(\log n)$ and bounded-fan gates). Indeed, letting $\mathbb{F} = \mathrm{GF}[2^{O(\log n)}]$ be a binary extension field, we can trivially implement field addition by a Boolean circuit of constant depth and $O(\log n)$ size (and bounded-fan gates). Field multiplication can be implemented by an $\mathrm{AC}^0[\oplus]$ circuit of size $\mathrm{polylog}(n)$ [HV06], and therefore by a Boolean circuit of size $\mathrm{polylog}(n)$, depth $\log(\mathrm{polylog}(n))$ and bounded-fan gates. It follows that L is in NC^1.

Finally, we note that in BGW addition gates require only local computation. This local computation consists of $O(n)$ parallel fan-in-2 additions of field elements. Since $\mathbb{F}$ is a binary extension field this can be implemented by a constant depth (NC^0) circuit of size $O(n \log n)$. We conclude that the extended view functionality g_π has the desired complexity. □

5 Degree-2 pMPRE for Protocol-Compatible Functionalities

In this section we show that any protocol-compatible functionality f can be encoded by a functionality $\hat{f}$ with *effective degree* 2. That is, each output of $\hat{f}$ can be computed as a degree-2 function over n values that can be computed by the parties locally (see Definition 2.1).

The following theorem will be proved in Section 5.1.

Theorem 5.1. *Let (f, P) be a protocol-compatible n-party functionality of depth d and output length m. Then, f has a perfect n-private MPRE $\hat{f}$ of effective-degree 2 and total complexity* $\mathrm{poly}(2^d, m)$.[6]

Remark 5.1 (Other properties of the MPRE)**.** *The encoding $\hat{f}$ constructed in Theorem 5.1 satisfies several additional properties that will not be used in our work, but may be useful elsewhere.*

1. *The encoding $\hat{f}$ is fully-decomposable and affine in x, that is for any fixing of the private randomness the residual functionality $\hat{f}(x)$ is a degree-1 function in x and each output bit of $\hat{f}$ depends on at most a single bit of the input x.*

2. *The preprocessing functions $(h_1, \ldots, h_n)$ that achieve effective degree of 2 only manipulate the private randomness. That is, we construct $h_i(x_i, r_i)$ s.t. $h_i(x_i, r_i) = (x_i, h'_i(r_i))$, where h'_i is a degree-2 function.*

5.1 Proof of Theorem 5.1

Let $f : X^n \to \{0,1\}^m$ be a protocol-compatible functionality of depth d. We now show how to encode f via a functionality

$$\hat{f} : (X \times R)^n \times R' \to Y'$$

with effective degree of 2. In addition to the private randomness r_i of each party, the functionality $\hat{f}$ uses internal randomness r'. (The latter can be removed via Proposition 3.2 while keeping an effective degree of 2.)

Notation. Let C be the Boolean circuit that represents f (as per Definition 4.2). Recall that the circuit C has m wires and it contains gates of two types: local computation gates and transmission (identity) gates. We prove the theorem with respect to circuits C in which the fan-out of transmission gates is one and the fan-out of local computation gates is two. This is without loss of generality, since any circuit C can be transformed to satisfy these restrictions while preserving the size (up to a constant factor), and at the expense of increasing the depth to $d' = d \log m$; we may ignore this overhead since $\mathrm{poly}(2^{d'}, m) = \mathrm{poly}(2^d, m)$. For every $i \in [m]$, let $P(i) \in [n]$ denote the party that holds the value of the ith wire in C.

Randomness. Our MPRE employs the following random bits. For every wire $i \in [m]$, the party $P(i)$ samples a random masking bit α_i. In addition, for every wire i, the functionality uses the internal randomness to sample a pair of random strings (keys) s_i^0, s_i^1 of length ω_i. The length ω_i of an "output wire" (i.e., a wire that does not enter any gate) is set to zero and the length of all other keys will be defined recursively (from top-to bottom) later. We assume that both strings, s_i^0, s_i^1, are partitioned to two equal-size blocks, and index these blocks by a bit $b \in \{0,1\}$, where $s_i^{a,b}$ denotes the bth block of s_i^a.

[6] Note that the circuit size of f does not appear explicitly in this statement, however for protocol-induced functionalities, the circuit size of f is equal to the output length m.

The outputs of the MPRE. We traverse the circuit C gate-by-gate in reverse topological order (from the output gates to the input wires), and let the functionality $\hat{f}$ deliver the following outputs to *all* parties.

- For every **local computation gate** g with incoming wires i, j and outgoing wires k, ℓ, we output four values (known as the gate table) defined as follows. For every $\beta_i, \beta_j \in \{0,1\}$, set

$$\gamma = G(\alpha_i \oplus \beta_i, \alpha_j \oplus \beta_j), \tag{2}$$

  where $G(\cdot,\cdot)$ is the function computed by the gate, and output the value

$$Q_g^{\beta_i,\beta_j} := \big((s_k^\gamma \| \gamma \oplus \alpha_k) \| (s_\ell^\gamma \| \gamma \oplus \alpha_\ell)\big) \oplus \; s_i^{\alpha_i \oplus \beta_i, \beta_j} \; \oplus \; s_j^{\alpha_j \oplus \beta_j, \beta_i} \; . \tag{3}$$

  One should view $Q_g^{\beta_i,\beta_j}$ as a ciphertext where the message is associated with the outgoing wires (first line of Eq. 3) is encrypted using a one-time pad under the combination of the keys associated with the incoming wires (second line of Eq. 3). Correspondingly, we set the length ω_i (resp., ω_j) of the keys s_i^0, s_i^1 (resp., s_j^0, s_j^1) to be $2(\omega_k + 1 + \omega_\ell + 1)$.

- **Transmission gates** are treated analogously. That is, for every transmission (identity) gate g with incoming wire i and outgoing wire k, we output the following two values. For every $\beta_i \in \{0,1\}$, set $\gamma = \beta_i \oplus \alpha_i$ and output the value

$$Q_g^{\beta_i} := (s_k^\gamma \| \gamma \oplus \alpha_k) \; \oplus \; s_i^{\alpha_i \oplus \beta_i} \; .$$

  Correspondingly, we set the length ω_i of the keys s_i^0, s_i^1 to be $\omega_k + 1$.

- For every **input wire** i, output the *masked value* $x_i \oplus \alpha_i$ and the *active key* $s_i^{x_i}$.

Effective degree and complexity. Observe that a term of the form s^a can be written as a degree-2 function of a and s (i.e., $a \cdot s^1 + (1-a) \cdot s^0$). Hence, all the outputs of the encoding are of degree 2 except for ciphertexts that correspond to local computation gates as in Eq. (3) in which the selection bit γ itself is a degree-2 function (and so the overall degree of $Q_g^{\beta_i,\beta_j}$ increases to 3). However, since the party $p = P(i) = P(j)$ knows both α_i and α_j, the value γ can be locally pre-computed and so the effective degree of the encoding is 2.

The complexity of the encoding is polynomial in the circuit size and the size of the largest key. A proof by induction shows that the length ω_i of the ith key is at most $O(4^{h_i})$ where h_i is the height of the ith wire (i.e., the length of the longest path from i to an "output wire" that does not enter any gate). Following this analysis, the complexity of $\hat{f}$ is bounded by $\mathrm{poly}(2^d, m)$.

Correctness. Fix some input $x = (x_1, \ldots x_n) \in X^n$ and let y_i denote the value induced by x on the ith wire. We show that the party $P(i)$ can recover y_i from the encoding $\hat{y}$ and its private randomness. Since the ith mask α_i is given to $P(i)$ as part of its private randomness, it suffices to show that $P(i)$ can recover the *masked value* $\hat{y}_i := y_i \oplus \alpha_i$. Indeed, as in standard garbled circuits, every party can recover the masked bit $\hat{y}_k := y_k \oplus \alpha_k$ together with the *active key* $s_k^{y_k}$, for every wire k. This is done by traversing the circuit from the inputs to the outputs as follows. For input wires the pair $\hat{y}_k, s_k^{y_k}$ is given explicitly as part of the encoding. For an internal wire k, that leaves a

local computation gate g with incoming wires i, j this is done by using the masked bits $\hat{y}_i, \hat{y}_j$ of the input wires to select the ciphertext $Q_k^{\hat{y}_i,\hat{y}_j}$ and then decrypting (i.e., XOR-ing) it with $s_i^{y_i,\hat{y}_j} \oplus s_j^{y_j,\hat{y}_i}$ that can be computed based on the active keys of the incoming wires. One can verify that this procedure recovers the desired values correctly. The case of transmission gates is treated similarly.

Privacy. We first claim that an external observer (that does not see the private randomness) can perfectly simulate the encoding given the list of masked values $(\hat{y}_k)_{k\in[m]}$.

Claim 5.2. *There exists a simulator* Sim' *that takes as an input an* m*-bit vector* $\hat{y} = (\hat{y}_i)_{i\in[m]}$*, runs in time* $\mathrm{poly}(m, 2^d)$ *and satisfies the following guarantee. For every input* x *and every fixing of* $\alpha = (\alpha_i)_{i\in[m]}$*, the random variable*

$$\mathsf{Sim}'(y_1 \oplus \alpha_1, \ldots, y_m \oplus \alpha_m),$$

where y_i *is the value induced by* x *on the* i*th wire, is distributed identically to the encoding* $\hat{f}(x)$ *conditioned on the above fixing of* α*.*

The claim is implicit in the standard proof of information-theoretic garbled circuit (cf. [IK02]); it is proved in Subsection 5.1.1 for completeness.

Based on Claim 5.2, we define a perfect simulator Sim for the MPRE. Fix an arbitrary coalition $T \subseteq [n]$ and let I be the set of wires owned by parties in T, i.e., $I = \{i : P(i) \in T\}$. Given the inputs $x[T]$ of T, and a vector of output values $(y_i)_{i\in I}$, the simulator does the following. For $i \in I$, sample uniformly the local randomness α_i and set $\hat{y}_i = y_i \oplus \alpha_i$. For $i \notin I$ sample $\hat{y}_i$ uniformly at random. Next invoke the simulator Sim' on $\hat{y} = (\hat{y}_i)_{i\in[m]}$ and output the result.

We prove that the simulation is perfect. Fix some input x, some $\alpha_I = (\alpha_i)_{i\in I}$, and let $y = f(x)$ and $y_I = (y_i)_{i\in I}$. We claim that the distribution sampled by $\mathsf{Sim}(T, x[T], \alpha_I, y_I)$ is identical to the joint distribution of the encoding $\hat{f}(x)$ induced by the choice of $\alpha_{[m]\setminus I}, (s_i^0, s_i^1)_{i\in[m]}$ (and conditioned on the above fixing of α_I). Indeed, since the marginal distribution of the vector of masked bits $(\hat{y}_1, \ldots, \hat{y}_m)$ is perfectly simulated, this follows from Claim 5.2.

5.1.1 Proof of Claim 5.2

Given $\hat{y}$ the simulator Sim' does the following.

1. Uniformly sample an active key $s_i^{y_i}$ for every wire $i \in [m]$. (Note that this can be done without knowing y_i.)

2. For every local computation gate g with incoming wires i, j and outgoing wires k, ℓ set the "active ciphertext"

$$Q_g^{\hat{y}_i,\hat{y}_j} := \big((s_k^{y_k} \| \hat{y}_k) \| (s_\ell^{y_\ell} \| \hat{y}_\ell)\big) \oplus s_i^{y_i,\hat{y}_j} \oplus s_j^{y_j,\hat{y}_i}.$$

   The other three "inactive" ciphertexts $Q_g^{a,b}$, $(a, b) \neq (\hat{y}_i, \hat{y}_j)$ are sampled uniformly at random.

3. For every transmission gate g with incoming wire i and outgoing wire k, set the "active ciphertext" to be

$$Q_g^{\hat{y}_i} := (s_k^{y_k} \| \hat{y}_k) \oplus s_i^{y_i}.$$

   Sample uniformly the other ciphertext $Q_g^{1\oplus\hat{y}_i}$.

4. For every input wire i output $s_i^{y_i}$ and $\hat{y}_i$.

Fix x, $\alpha = (\alpha_i)_{i\in[m]}$ and let $(y_1, \ldots, y_m) = f(x)$ and $\hat{y}_i = y_i \oplus \alpha_i$. We prove that the simulator satisfies the claim. First observe that the marginal distribution of the simulated active keys is identical to their distribution in the encoding. Next observe that the value of all active ciphertexts is fully determined by α and by the value of the active keys, and it is computed in the simulation just like it is in the encoding. To complete the analysis, it suffices to show that, conditioned on any fixing of the active keys and α, in the encoding $\hat{f}(x)$ the inactive ciphertexts of every gate are distributed uniformly and independently. To this end, we observe that for every inactive ciphertext Q we can associate a unique inactive (sub-)key s so that $Q = h \oplus s$ where h is independent of s (and may depend on all other inputs/randomness). Indeed, for a transmission gate g with incoming wire i, the inactive ciphertext $Q_g^{1\oplus\hat{y}_i}$ is associated with $s_i^{1\oplus y_i}$. For a local computation gate g with incoming wires i, j, the inactive ciphertext $Q_g^{a,b}$ for $(a, b) \neq (\hat{y}_i, \hat{y}_j)$ is associated with $s_i^a[b]$ if $a \neq \hat{y}_i$ and with $s_j^b[a]$, otherwise (note that in this case $b \neq \hat{y}_j$). This mapping from inactive ciphertexts to inactive keys satisfies the above requirements. □

6 Degree-2 cMPRE for Protocol-Compatible Functionalities

Theorem 5.1 has an exponential dependency on the depth of the encoded function. In this section we show that this dependency can be removed at the expense of relaxing privacy to computational. The transformation is based on computationally-private garbled circuits and is closely related to the construction of [BMR90].

Theorem 6.1. *Assuming the existence of one-way functions, any protocol-compatible n-party functionality (f, P) with output length m has a computational n-private MPRE $\hat{f}$ of effective-degree 2 and total complexity* $\mathrm{poly}(m, \lambda)$*, where λ is the security parameter.*[7]

Moreover, the one-way function is only used in the preprocessing phase and in a black-box manner and the properties of Remark 5.1 apply here as well.

6.1 Proof of Theorem 6.1

The setting is similar to Section 5.1, but we do not use internal randomness. Let $f : X^n \to \{0,1\}^m$ be a protocol-compatible functionality. We now show how to encode f via a functionality

$$\hat{f} : (X \times R)^n \to Y'$$

with effective degree of 2. We use the same notational conventions with respect to the topology of the circuit as in Section 5.1, where internal gates have two inputs and two outputs, and transmission gates have one input and one output. Recall that we define $P : [m] \to [n]$ s.t. $P(i)$ denotes the party that holds the value of the ith wire in C.

Let λ denote the security parameter and let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{4n\lambda+2}$ be a pseudorandom generator. If one-way functions exist, then such PRG exists as well, and computing PRG requires only black-box use of the one-way function.

[7] As in Theorem 5.1, the circuit size of f is equal to the output length m and thus it does not appear explicitly.

Randomness. Our MPRE employs the following random bits. For every wire $i \in [m]$, the party $P(i)$ samples a random masking bit α_i. In addition, for every wire i, every party ν samples four random strings denoted by $s_{i,\nu}^{b,c} \in \{0,1\}^\lambda$ for $b, c \in \{0,1\}$. For $b \in \{0,1\}$, the string $s_{i,\nu}^b = (s_{i,\nu}^{b,0} \| s_{i,\nu}^{b,1}) \in \{0,1\}^{2\lambda}$ is viewed as the ν-share of the b-key of wire i. Party ν then computes, as a preprocessing step, the values $w_{i,\nu}^{b,c} = \mathsf{PRG}(s_{i,\nu}^{b,c}) \in \{0,1\}^{4n\lambda+2}$ for all i, b and c. Note that this only requires black-box use of PRG. We let $S_i^b \in \{0,1\}^{2n\lambda}$ denote the b-th key of wire i, and define it to be the in-order concatenation of the strings $(s_{i,\nu}^b)_{\nu=1}^n$. For $b, c \in \{0,1\}$, denote $W_i^{b,c} = \bigoplus_{\nu=1}^n w_{i,\nu}^{b,c} \in \{0,1\}^{4n\lambda+2}$, and furthermore let us define $W_i^b = W_i^{b,0} \| W_i^{b,1}$ to be the b-th pad of wire i.

The outputs of the MPRE. We traverse the circuit C gate-by-gate in reverse topological order (from the output gates to the input wires), and let the functionality $\hat{f}$ deliver the following outputs to *all* parties.

- For every **local computation gate** g with incoming wires i, j and outgoing wires k, ℓ, we output four values (known as the gate table) defined as follows. For every $\beta_i, \beta_j \in \{0,1\}$, set
$$\gamma = G(\alpha_i \oplus \beta_i, \alpha_j \oplus \beta_j), \tag{4}$$
where $G(\cdot,\cdot)$ is the function computed by the gate, and output the value
$$Q_g^{\beta_i,\beta_j} := \big((S_k^\gamma \| \gamma \oplus \alpha_k) \| (S_\ell^\gamma \| \gamma \oplus \alpha_\ell)\big) \oplus W_i^{\alpha_i \oplus \beta_i, \beta_j} \oplus W_j^{\alpha_j \oplus \beta_j, \beta_i} . \tag{5}$$
As in Section 5.1, one should view $Q_g^{\beta_i,\beta_j}$ as a ciphertext where the message is associated with the seeds of the outgoing wires (first line of Eq. 5), and it is encrypted using a mask which is the XOR of PRG outputs of all parties on keys associated with the incoming wires (second line of Eq. 5).
- **Transmission gates** are treated analogously. That is, for every transmission (identity) gate g with incoming wire i and outgoing wire k, we output the following two values. For every $\beta_i \in \{0,1\}$, set $\gamma = \beta_i \oplus \alpha_i$ and output the value
$$Q_g^{\beta_i} := (S_k^\gamma \| \gamma \oplus \alpha_k \| 0^{2n\lambda+1}) \oplus W_i^{\alpha_i \oplus \beta_i, 0} ,$$
where the zero padding is only a matter of formality to match the length of the W_i string.
- For every **input wire** i, output the *masked value* $x_i \oplus \alpha_i$ and the *active key set* $S_i^{x_i}$.

Effective degree and complexity. This is again very similar to the information theoretic construction. Observe that a terms of the form S^a, W^a can be written as a degree-2 function of a and S, W respectively, and that S, W themselves are linear functions in the precomputed strings s, w. Hence, all the outputs of the encoding are of degree 2 except for ciphertexts that correspond to local computation gates as in Eq. (5) in which the selection bit γ itself is a degree-2 function (and so the overall degree of $Q_g^{\beta_i,\beta_j}$ increases to 3). However, since the party $p = P(i) = P(j)$ knows both α_i and α_j, the value γ can be locally pre-computed and so the effective degree of the encoding is 2.

The complexity of the encoding is polynomial in the circuit size, as any gate requires a gadget of length $O(n\lambda)$. The complexity of $\hat{f}$ is therefore $O(mn\lambda)$.

Analysis. Both correctness and privacy can be analyzed directly following standard proofs of garbled circuits. Instead, we show here how to deduce the MPRE analysis from the analysis of the GC-based randomized encoding from [AIK06]. Following Remark 3.3, it suffices to prove that for any coalition $T \subseteq [n]$ the T-restricted function f_T is encoded by the T-restricted randomized function $\hat{f}_T$. Recall that $f_T(x)$ denotes the function that computes the functionality f and outputs all the values that are held by the parties in T, and that $\hat{f}_T(x)$ is a randomized function whose internal randomness consists of all the (private) randomness used in $\hat{f}$ and its output consists of the output of $\hat{f}(x)$ concatenated with the internal randomness of the players in T. Formally, let

$$f_T(x) := ((x_i)_{i\in T}, (y_i)_{i\in I}) \qquad \text{where } y = f(x) \text{ and } I = \{i : P(i) \in T\}$$

and let

$$\hat{f}_T(x; s_T, s_{\overline{T}}, \alpha_T, \alpha_{\overline{T}}) = ((x_i)_{i\in T}, \hat{f}(x; s_T, s_{\overline{T}}, \alpha_T, \alpha_{\overline{T}}), s_T, \alpha_T),$$

where $s_T := (s^0_{i,\nu}, s^1_{i,\nu})_{i\in[m],\nu\in T}$, $s_{\overline{T}} := (s^0_{i,\nu}, s^1_{i,\nu})_{i\in[m],\nu\notin T}$, $\alpha_T := (\alpha_i)_{i\in I}$ and $\alpha_{\overline{T}} := (\alpha_i)_{i\notin I}$.

By abuse of notation, we identify f_T with the same circuit that computes f except that the output wires are taken to be all the wires whose index is in I and all the input wires that are held by T (that correspond to x_T).

It suffices to prove the following claim:

Claim 6.2. *The randomized function $\hat{f}_T$ is a (standard) computationally-private perfectly-correct randomized encoding of f_T. Moreover, there exists an efficient algorithm that given the circuit f, and security parameter 1^λ, outputs a pair of circuits $\mathsf{Dec}_\lambda(\cdot,\cdot)$ and $\mathsf{Sim}_\lambda(\cdot,\cdot)$ such that $(\mathsf{Dec}_\lambda(T,\cdot), \mathsf{Sim}_\lambda(T,\cdot))$ is a decoder/simulator pair for the encoding $\hat{f}_T$.*

Proof. We prove a stronger claim, namely, that $\hat{f}_T$ encodes f_T even when s_T is arbitrarily fixed.[8] Fix s_T and let $n' = |\bar{T}|$ denote the number of uncorrupted players. Assume that $n' > 0$ (i.e., that T is a strict subse of $[n]$), since otherwise the claim trivially holds. We show that, when viewed correctly, $\hat{f}_T$ simplifies into the construction of computational randomized encoding (CRE) from [AIK06]. First, observe that we can remove the entries $(x_i)_{i\in T}$ from the output of $\hat{f}_T$ since each such x_i can be derived from α_i and from $x_i \oplus \alpha_i$ which are already present in the encoding. Next, we describe the CRE of [AIK06] and explain its correspondence to $\hat{f}_T$:

- Randomness: For each wire i, the CRE samples a mask bit, together with a zero-key S^0_i and a one-key S^1_i. Each of these keys consists of two equal-length sub-keys, i.e., $S^b_i = (S^{b,0}_i \| S^{b,1}_i)$, for $b \in \{0,1\}$. In our context, we let α_i take the role of the mask bit and let $S^{b,c}_i \in \{0,1\}^{n'\lambda}$ denote the in-order concatenation of the strings $(s^{b,c}_{i,\nu})_{\nu\in\bar{T}}$. (Note that unlike the notation used in the construction, here $S^{b,c}_i$ consists only of the keys that are being held by the uncorrupted parties.)

- For each input wire i the encoding outputs the masked value $x_i \oplus \alpha_i$ and the active key set $S^{x_i}_i$. These values are also outputted by $\hat{f}_T$.

- For each output wire i the encoding outputs the mask α_i. These are exactly the values $(\alpha_i)_{i\in I}$ that are outputted by $\hat{f}_T$ (since we defined the set of output wires to be the set of wires held by the partes in T).

[8] One can further show that this is true even when both s_T and α_T are arbitrarily fixed. We omit the proof since it will not be needed for our results.

- In addition, for each gate g with incoming wires i, j and outgoing wires k, ℓ, the encoding outputs four ciphertexts $(C_g^{0,0}, C_g^{0,1}, C_g^{1,0}, C_g^{1,1})$. For $\beta_i, \beta_j \in \{0,1\}$ we let $C_g^{\beta_i,\beta_j} = E_K(M)$, where
$$M = \left((S_k^\gamma \| \gamma \oplus \alpha_k) \| (S_\ell^\gamma \| \gamma \oplus \alpha_\ell)\right), \qquad K = S_i^{\alpha_i \oplus \beta_i, \beta_j} \oplus S_j^{\alpha_j \oplus \beta_j, \beta_i}, \tag{6}$$
and γ is defined as in (4).

It is shown in [AIK06] that if E is a (possibly randomized) one-time computationally-secure encryption scheme that admits perfectly-correct decryption algorithm D, the resulting encoding is a CRE of the function f_T. Hence, to complete the proof it suffices to show that the expression $Q_g^{\beta_i,\beta_j}$ defined in (5) can be written as a ciphertext of the form $C_g^{\beta_i,\beta_j}$. Indeed, letting M and K be defined as in (6), we can write $Q_g^{\beta_i,\beta_j}$ (possibly after reordering its coordinates) as

$$E_K(M) := (M \| M') \oplus \mathsf{PRG}'(K) \oplus K',$$

where M' and K' are constants that depend on the constant s_T, and PRG' parses its seed $K \in \{0,1\}^{n'\lambda}$ into n' blocks $(K_1, \ldots, K_{n'})$ of length λ each, and outputs $\mathsf{PRG}(K_1) \oplus \cdots \oplus \mathsf{PRG}(K_{n'}))$. Observe that PRG' is a pseudorandom generator, and therefore the encryption function E is one-time semantically-secure (with a straightforward decryption algorithm). We conclude that $\hat{f}_T$ fits the construction from [AIK06] and is itherefore a CRE of f_T.

Finally, the simulator and decoder of the CRE of [AIK06] are efficiently constructible by a compiler A that takes 1^λ and the circuit f_T as inputs. We can therefore define an efficient algorithm A' that, given 1^λ, the circuit f, and (a representation of) a set $T \subset [n]$, (1) computes the circuit f_T, (2) applies A to this circuit, and (3) outputs a decoder and simulator circuits for $\hat{f}_T$. By embedding steps (1–3) of the algorithm into the decoder and simulator circuits, we derive the "moreover" statement. □

7 Putting It All Together

In this section we prove the following theorems using the tools we developed in previous sections.

Theorem 7.1. *Every n-party functionality f can be encoded by a perfect MPRE g with privacy threshold of $\tau = \left\lfloor \frac{n-1}{2} \right\rfloor$, effective degree 2 and complexity polynomial in n and S where S is the size of the branching program that computes f.*

Theorem 7.2. *Assuming the existence of one-way functions, every n-party functionality f can be encoded by a computational MPRE g with privacy threshold of $\tau = \left\lfloor \frac{n-1}{2} \right\rfloor$, effective degree 2 and complexity polynomial in n and S where S is the size of the circuit that computes f. Moreover, the MPRE makes use of one-way functions in a black-box way only as part of the local preprocessing step.*

Theorem 7.3. *Assuming the existence of (possibly multi-round) oblivious transfer, every n-party functionality f can be encoded by a computational MPRE g with privacy threshold of $\tau = n$, effective degree 2 and complexity polynomial in n and S where S is the size of the circuit that computes f. The MPRE makes a non-black-box use of the oblivious transfer protocol.*

Theorems 7.1, 7.2 and 7.3 (whose proofs are deferred to Sections 7.1, 7.2 and 7.3) can be used to derive our main results (Theorems 1.1, and 1.2).

Proof of Theorem 1.1 and 1.2. We prove Theorem 1.1 (resp., Theorem 1.2): Given an n-party functionality f that is computable by a branching program of size S (resp., computable by a Boolean circuit of size S), construct the perfect MPRE g promised by Theorem 7.1 (resp., the computational MPRE g promised by Theorem 7.2). By Proposition 3.1, f non-interactively $\lfloor \frac{n-1}{2} \rfloor$-reduces to g with perfect privacy (resp., computational privacy). Since g has an effective degree 2, the functionality g itself n-privately reduces to a degree-2 functionality g' (in a trivial way). A composition of these reductions yields the desired reduction.

To prove the second ("Consequently") part of the theorem, we employ the BGW protocol $\pi_{g'}$ to privately compute g' in 2 rounds (since its degree is 2) and complexity of $\text{poly}(n, S)$ at the presence of honest majority. Plugging this protocol into the above reduction and using standard composition theorems (cf. [Can00]), we get a 2-round protocol for f with similar complexity and perfect (resp., computational) privacy. □

To prove our results in the client-server model (Theorem 1.3) we will need the following lemma whose proof is deferred to Section 7.4.

Lemma 7.4. *Let f be an n-party functionality and let g be a perfectly-correct MPRE for f with effective degree-2, complexity T, privacy threshold τ and perfect privacy (resp., computational privacy). Then, f can be computed in the client-server model in two rounds with n clients, an arbitrary number m of servers, and perfect (respectively, computational) privacy against any adversary that corrupts a minority of the servers and at most τ-fraction of the clients. The complexity of the protocol is polynomial in n, m and T.*

Theorem 1.3 follows immediately by combining Theorems 7.1, 7.2 and 7.3 with Lemma 7.4.

7.1 Perfect MPRE for Branching Programs (Proof of Theorem 7.1)

Let f be an n-party functionality that is computable by a branching program of size S. By [IK00], such a function has degree-3 perfect randomized encoding $g_1(x; r)$ of $\text{poly}(S)$ size. Recall that such an RE yields an n-private MPRE, and let us get rid of the private randomness by applying Proposition 3.2. This gives us a degree-3 MPRE g_2 of f whose complexity is $\text{poly}(S)$ with privacy threshold of n. Next, we encode g_2 by the BGW-based protocol-compatible encoding (Theorem 4.3) and get a protocol-compatible perfect MPRE g_3 of size $O(S \cdot \text{poly}(n))$, depth $O(\log n)$ and privacy threshold of $\tau = \lfloor \frac{n-1}{2} \rfloor$. Using our information-theoretic encoding from Theorem 5.1 (based on garbled circuits), we get a τ-private perfect MPRE g_4 of g_3 with complexity $\text{poly}(n, S)$ and effective degree 2. By the composition lemma (Lemma 3.3), the MPRE g_4 perfectly encode f with privacy threshold of τ. □

7.2 Computational MPRE for Circuits (Proof of Theorem 7.2)

Let f be an n-party functionality that is computable by a circuit of size S. By Theorem 4.3, we can apply the BGW-based protocol-compatible encoding (Theorem 4.3) and get a protocol-compatible perfect MPRE g_1 of size $O(S \cdot \text{poly}(n))$, and privacy threshold of $\tau = \lfloor \frac{n-1}{2} \rfloor$. Using our computational encoding from Theorem 6.1 (based on garbled circuits), we get a τ-private computational MPRE g_2 of g_1. This MPRE has complexity $\text{poly}(n, S)$, effective degree 2, and

it makes a black-box use of the one-way function. By the composition lemma (Lemma 3.3), the MPRE g_2 computationally encode f with privacy threshold of τ.[9] □

7.3 Computational n-private MPRE for Circuits (Proof of Theorem 7.3)

Let f be an n-party functionality that is computable by a circuit of size S. Assuming the existence of oblivious transfer, the GMW protocol [GMW87] computes f with computational privacy, complexity of $\text{poly}(n, S)$, and privacy threshold of $\tau = n$. By applying Proposition 4.1, we derive a GMW-based protocol-compatible encoding g_1 of circuit size $\text{poly}(n, S)$, computational privacy, and privacy threshold of τ.[10] Since the existence of oblivious transfer implies the existence of one-way functions, we can use the computational encoding from Theorem 6.1 (based on garbled circuits), and get a τ-private computational MPRE g_2 of g_1 with complexity $\text{poly}(n, S)$, and effective degree 2. By the composition lemma (Lemma 3.3), the MPRE g_2 computationally encode f with privacy threshold of τ. □

7.4 Client-Server Protocols for MPRE with effective degree-2 (Proof of Lem. 7.4)

We need the following fact that is based on a client-server variant of the standard protocol of [BGW88, CCD88]. (A related observation was made in [IKP10].)

Fact 7.5. *Any n-party degree-2 functionality q can be perfectly computed with n clients an arbitrary number of m servers and perfect security against any semi-honest adversary that corrupts a minority of the servers and any subset of the clients.*

sketch. We sketch the protocol for completeness, focusing on the case of a deterministic functionality q that delivers a single output to a single party. (Multi-output functionalities can be handled by repeating the above process in parallel, and randomized functionalities reduces to deterministic ones via standard degree-preserving reduction; cf. Proposition 3.2.)

Let $\mathbb{F}$ be the field $\text{GF}(2^{\lceil \log m \rceil})$. For each input x_i the corresponding party selects a random degree-$\lfloor m/2 \rfloor$ polynomial P_i over $\mathbb{F}$ whose free coefficient equals to x_i and sends $x_{i,j} = P_i(j)$ to the j-th server. In addition, each party i selects a random degree-m polynomial R_i and sends to the j-the server the share $r_{i,j} = R_i(j)$. The j-th server locally computes the output of the degree-2 functionality on the shares $q(x_{1,j}, \ldots, x_{n,j})$, adds the zero shares $r_{1,j} + \ldots + r_{n,j}$, and sends back the result to the client who owns the output. This client interpolates the received points and outputs the free coefficient of the corresponding degree-m polynomial. The analysis of the protocol follows the standard analysis of the BGW/CCD protocol [BGW88, CCD88]. □

Let

$$g((x_1, r_1), \ldots, (x_n, r_n), r_0) = q(h_1(x_1, r_1), \ldots, (x_n, r_n); r_0)$$

be the MPRE for f where h_i denotes the preprocessing function of the i-th party and q is a degree-2 functionality. We use the protocol π promised by Fact 7.5 to securely compute g and show that the resulting protocol realizes f. (This is essentially similar to the reduction proved in Prop. 3.1, except that here we are dealing with the client-server model.)

The protocol for f proceeds as follows.

[9] One can obtain an alternative proof that does not rely on Theorem 6.1, and instead employs the constant-degree MPRE that is induced by the constant-round multiparty protocol of [BMR90, Rog91]. (See the conference version of this paper for details.)

[10] This MPRE makes a non-black box use of the oblivious transfer that is employed by the GMW protocol.

- Each client i samples its local randomness r_i for the MPRE g and locally preprocess its input (x_i, r_i) into $z_i = h_i(x_i, r_i)$.
- The parties invoke the client-server protocol π for the randomized functionality q with the inputs z_i, and apply their decoders on the result.

The complexity of the protocol is polynomial in n, m and the complexity of the MPRE g. We prove that the protocol privately realizes f. By Proposition 3.2, we may assume, without loss of generality, that the functionality f is a deterministic functionality. In this case, it suffices to establish correctness and privacy separately (cf. [Gol04, Section 7.2.2]).

Correctness follows immediately from the correctness of π and the correctness of the MPRE. (Just like in the proof of Proposition 3.1.) Privacy follows by composing the simulators of the protocol π with the MPRE simulator. Formally, consider an adversary that corrupts a τ-subset T_1 of the clients and a minority T_2 of the servers. Let Sim_π be the simulator of the protocol π that takes as an input the set of corrupted parties (T_1, T_2), the inputs of the corrupted clients $z[T_1]$, and their output $\hat{y}[T_1]$ and perfectly samples the joint view of the corrupted parties (in T_1 and T_2). Let Sim_g be the simulator of the MPRE that takes as an input the set of corrupted parties T_1, their inputs $x[T_1]$ and their final outputs $y[T_1]$, and samples perfectly (resp., computationally) the joint MPRE view $(x[T_1], r[T_1], \hat{y}[T_1])$. We define the simulator $\mathsf{Sim}(T_1, T_2, x[T_1], y[T_1])$ as follows: (1) Apply $\mathsf{Sim}_g(T_1, x[T_1], y[T_1])$ and get $(x[T_1], r[T_1], \hat{y}[T_1])$; (2) Compute $z_i = h_i(x_i, r_i)$ for every $i \in T_1$; (3) Compute the π-view $\mathsf{Sim}_\pi((T_1, T_2), z[T_1], \hat{y}[T_1])$ of the parties in T_1 and T_2, and output the result concatenated with $(x[T], r[T])$.

The perfect privacy (resp., computational) of the simulator follows from the perfect privacy of π and the perfect (resp., computational) privacy of the MPRE. This completes the proof of the lemma. □

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, pages 152–174, 2018.

[ABT19] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 504–531. Springer, 2019.

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 395–424, 2018.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175. IEEE Computer Society, 2004.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography*, pages 1–44, 2017.

[BB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*, pages 201–209. ACM, 1989.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Simon [Sim88], pages 1–10.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Nielsen and Rijmen [NR18], pages 500–532.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Simon [Sim88], pages 11–19.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 378–394, 2005.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994.

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: information-theoretic and black-box. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, pages 123–151, 2018.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 588–599, 2017.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Nielsen and Rijmen [NR18], pages 468–499.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 672–683. Springer, 2006.

[IK00] 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 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304. IEEE Computer Society, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2010.

[Ish13] Yuval Ishai. Randomization techniques for secure computation. In Manoj Prabhakaran and Amit Sahai, editors, *Secure Multi-Party Computation*, volume 10 of *Cryptology and Information Security Series*, pages 222–248. IOS Press, 2013.

[NR18] Jesper Buus Nielsen and Vincent Rijmen, editors. *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018.

[Rog91] Philip Rogaway. *The Round-Complexity of Secure Protocols*. PhD thesis, MIT, 1991.

[Sim88] Janos Simon, editor. *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. ACM, 1988.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.