


# Asymmetric Multi-Party Computation

Vipul Goyal ✉

NTT Research, Sunnyvale, CA, USA  
Carnegie Mellon University, Pittsburgh, PA, USA

Chen-Da Liu-Zhang ✉ 

NTT Research, Sunnyvale, CA, USA

Rafail Ostrovsky ✉

University of California at Los Angeles, CA, USA

---

## Abstract

Current protocols for Multi-Party Computation (MPC) consider the setting where all parties have access to similar resources. For example, all parties have access to channels bounded by the same worst-case delay upper bound  $\Delta$ , and all channels have the same cost of communication. As a consequence, the overall protocol performance (resp. the communication cost) may be heavily affected by the slowest (resp. the most expensive) channel, even when most channels are fast (resp. cheap). Given the state of affairs, we initiate a systematic study of *asymmetric* MPC. In asymmetric MPC, the parties are divided into two categories: fast and slow parties, depending on whether they have access to high-end or low-end resources.

We investigate two different models. In the first, we consider asymmetric communication delays: Fast parties are connected via channels with small delay  $\delta$  among themselves, while channels connected to (at least) one slow party have a large delay  $\Delta \gg \delta$ . In the second model, we consider asymmetric communication costs: Fast parties benefit from channels with cheap communication, while channels connected to a slow party have an expensive communication. We provide a wide range of positive and negative results exploring the trade-offs between the achievable number of tolerated corruptions  $t$  and slow parties  $s$ , versus the round complexity and communication cost in each of the models. Among others, we achieve the following results. In the model with asymmetric communication delays, focusing on the information-theoretic (i-t) setting:

- An i-t asymmetric MPC protocol with security with abort as long as  $t + s < n$  and  $t < n/2$ , in a constant number of slow rounds.
- We show that achieving an i-t asymmetric MPC protocol for  $t + s = n$  and with number of slow rounds independent of the circuit size implies an i-t synchronous MPC protocol with round complexity independent of the circuit size, which is a major problem in the field of round-complexity of MPC.
- We identify a new primitive, *asymmetric broadcast*, that allows to consistently distribute a value among the fast parties, and at a later time the same value to slow parties. We completely characterize the feasibility of asymmetric broadcast by showing that it is possible if and only if  $2t + s < n$ .
- An i-t asymmetric MPC protocol with guaranteed output delivery as long as  $t + s < n$  and  $t < n/2$ , in a number of slow rounds independent of the circuit size.

In the model with asymmetric communication cost, we achieve an asymmetric MPC protocol for security with abort for  $t + s < n$  and  $t < n/2$ , based on one-way functions (OWF). The protocol communicates a number of bits over expensive channels that is independent of the circuit size. We conjecture that assuming OWF is needed and further provide a partial result in this direction.

**2012 ACM Subject Classification** Security and privacy → Cryptography

**Keywords and phrases** multiparty computation, asymmetric, delays, communication

**Digital Object Identifier** 10.4230/LIPIcs.ITC.2023.6



© Vipul Goyal, Chen-Da Liu-Zhang, and Rafail Ostrovsky;  
licensed under Creative Commons License CC-BY 4.0

4th Conference on Information-Theoretic Cryptography (ITC 2023).

Editor: Kai-Min Chung; Article No. 6; pp. 6:1–6:25



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Funding** *Rafail Ostrovsky*: The author was supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, the Algorand Centers of Excellence programme managed by Algorand Foundation, NSF grants CNS-2246355, CCF-2220450 and CNS-2001096, US-Israel BSF grant 2015782, Amazon Faculty Award, Cisco Research Award and Sunday Group. Any views, opinions, findings, conclusions or recommendations contained herein are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, the Algorand Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.

## 1 Introduction

Secure Multi-Party Computation [49, 24, 7, 11, 46] allows a set of distrustful parties to compute a function over their private inputs, in such a way that nothing about the inputs is revealed beyond the output of the computation.

Generally speaking, current MPC protocols consider the simplest setting where all parties have network resources with the same guarantees. In particular, the most common *synchronous* network model considers the setting where all channel delays are upper bounded by a single worst-case delay  $\Delta$  and all channels have the same cost of communication. Even though this model is theoretically interesting, it suffers from important practical limitations. In particular,  $\Delta$  has to be set large enough to accommodate any possible delay: Even in cases where almost all parties have fast channels with delay  $\delta \ll \Delta$ , the protocols do not take advantage of this, and the running time of the protocol is affected by the slowest party. This is particularly critical for information-theoretic protocols, where all current solutions have a round complexity that depends on the depth of the circuit to evaluate. Similarly, the protocols designed in this model also fail to take advantage of the cost of communication from channels that are cheap, and the total communication cost is affected by the most expensive channel.

Given the state of affairs, we initiate the study of *asymmetric* MPC. In asymmetric MPC, the parties are divided into two categories. We consider *fast* parties, which are parties that have access to high-end network resources (e.g. channels with small delay, cheap channels, etc), and *slow* parties, which are parties that have access only to low-end network resources. One can think about the fast parties as parties that are in some sense privileged and have access to fast and cheap internet connection, e.g. with fiber, while slow parties are not so privileged, and only have access to slow and expensive connectivity, e.g. one may think of mobile devices or IoT devices such as sensors collecting data. Considering parties with asymmetric resources allows us to not only model more realistic scenarios, where parties have access to different levels of resources, but also to design more refined protocols that exploit such asymmetries, thereby improving the performance and communication cost of protocols, while at the same time achieving more refined levels of security and assumptions.

## 2 Our Contributions

We initiate a systematic study of asymmetric MPC with respect to two different models for network resources. In the first model, we consider a network with asymmetric communication delay. Fast parties are connected via channels with small delay  $\delta$  among themselves, while all other channels, which are connected to a slow party, have a large delay  $\Delta$ . In the second

model, we consider a network with asymmetric communication cost. This means that fast parties benefit from channels with cheap communication, while other channels incur a high cost of communication.

Our focus is on minimizing the complexity from the slow parties (minimizing the number of slow rounds, respectively the usage of expensive channels) while at the same time tolerating as many corruptions and slow parties as possible. This allows us to give a first overall study of asymmetric MPC protocols in a clean manner.

*What are the achievable trade-offs between the number of tolerated corruptions and slow parties compared to the number of slow rounds in the model with asymmetric delays? And similarly, trade-offs with respect to the number of bits transmitted over expensive channels in the model with asymmetric communication cost?*

To the best of our knowledge, no previous work has addressed the setting of asymmetric MPC, for any of the resource models. For example, synchronous protocols assume that all channels have the same worst-case delay and also that the cost of communication is the same for all channels. Similarly, asynchronous protocols assume that all channels have *eventual* delivery, and that all channels have the same cost of communication.

Note, however, that existing protocols using standard cryptographic assumptions do achieve a constant number of slow rounds [49, 5], and FHE-based protocols [40, 1, 26] achieve low communication over expensive channels, so our focus is on information-theoretic protocols when minimizing the number of slow rounds, and non-FHE protocols when minimizing the communication over expensive channels. See a more detailed discussion in Appendix A.

## 2.1 MPC with Asymmetric Delays

**The Model.** As mentioned above, in this model we divide the parties into two categories: fast and slow parties. Parties have access to a complete network of point-to-point (P2P) channels. The channels between fast parties have a small delay  $\delta$  and are denoted *fast channels*, and all the other channels, i.e. the channels that contain at least one slow party, have a large delay  $\Delta$  and are denoted *slow channels*.

We are interested in counting the number of slow P2P-rounds, which are the number of P2P communication steps via all channels, and the number of fast rounds, which are the number of P2P communication steps via the fast channels. Optimally, we would like to find protocols that have a constant number of slow P2P-rounds, or at least a number of slow P2P-rounds that is independent of the circuit to evaluate. Throughout the paper, we will omit the P2P term, and simply denote such rounds as fast and slow rounds.

**Existing Solutions.** Current constant-round solutions based on cryptographic assumptions [49, 5, 37, 35, 42, 34, 20, 8], are already asymmetric MPC protocols with a constant number of slow rounds. This is because any constant-round synchronous MPC protocol trivially implies an asymmetric MPC protocol in our setting with a constant number of slow rounds.

**Information-Theoretic Protocols.** Information-theoretic protocols are much more interesting, since all current synchronous solutions require a number of rounds proportional to the depth of the circuit to evaluate. This is in stark contrast with protocols in our model, where we will be able to achieve a number of slow rounds that is independent of the circuit (sometimes even constant). We propose several information-theoretic protocols for the setting of malicious security, with abort and with guaranteed output delivery.

In the following, we let  $C$  be the circuit to evaluate, and let  $n, s, t$  be the number of parties, bound on the number of slow parties, and bound on the number of corruptions.

**Security with Abort.** We first present a protocol that achieves security with abort and is secure as long as  $t + s < n$  and  $t < n/2$ . The round complexity is  $O(1)$  slow rounds and  $O(\text{depth}(C))$  fast rounds.

► **Theorem 1.** *Let  $n, s, t$  be natural numbers such that  $t + s < n$  and  $t < n/2$ . There is an information-theoretic asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with security with abort, in the presence of up to  $t$  malicious corruptions and  $s$  slow parties. The round complexity is  $O(1)$  slow rounds and  $O(\text{depth}(C))$  fast rounds.*

We then show that improving the resiliency of our asymmetric MPC protocol requires a breakthrough. Assume that  $t + s = n$ . Then, the following result implies that information-theoretic asymmetric MPC in a constant number of slow rounds implies constant-round information-theoretic MPC in the synchronous model resilient up to 1 corruption, which is known to be a major barrier in information-theoretic MPC.

► **Theorem 2.** *Let  $n, s, t > 0$  be natural numbers such that  $t + s = n$ . Then  $n$ -party information-theoretic asymmetric MPC with security with abort (resp. guaranteed output delivery), resilient up to  $t$  corruptions and  $s$  slow parties in  $R$  slow rounds, implies  $(s + 1)$ -party information-theoretic synchronous MPC with security with abort (resp. guaranteed output delivery), resilient up to 1 corruption in  $R$  rounds.*

**Guaranteed Output Delivery.** In the setting of malicious security with guaranteed output delivery, we present two results.

A Protocol for  $2t + s < n$  and  $t < n/3$ . We start by presenting a solution for an information-theoretic protocol with guaranteed output delivery in the regime where  $2t + s < n$  and  $t < n/3$  (with no setup nor broadcast).

► **Theorem 3.** *Let  $\kappa$  be a security parameter. Let  $n, s, t$  be natural numbers such that  $2t + s < n$  and  $t < n/3$ . There is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with guaranteed output delivery in the presence of up to  $t$  malicious corruptions and  $s$  slow parties. The round complexity is  $O(\kappa)$  slow rounds and  $O(\text{depth}(C) \cdot \kappa)$  fast rounds.*

In the above theorem statement, if synchronous broadcast channels are assumed as setup (or alternatively, a setup for i-t signatures [44]), the condition  $t < n/3$  is not necessary. The above protocol inherently requires the condition  $2t + s < n$ . However, optimally one would wish to require an honest majority *overall*, rather than among the fast parties. We therefore want to find protocols that deal with a dishonest majority among the fast parties.

*Broadcast with Asymmetric Delays.* To overcome this bound, we identify a natural primitive in our setting with asymmetric delays, called *asymmetric broadcast*. This primitive ensures that all the fast parties obtain the output within  $d_{bc}$  fast rounds, while slow parties obtain the same output much later, within  $D_{bc}$  slow rounds. We call the quantity  $d_{bc}$  the fast asymmetric broadcast delay, and  $D_{bc}$  the slow asymmetric broadcast delay.

Our first step is to investigate the possible trade-offs for asymmetric broadcast. Our results completely characterize the feasibility of asymmetric broadcast from point-to-point channels, by showing matching positive and negative results.

First, in Section 5.2, we show a simple construction of  $n$ -party asymmetric broadcast with a fast sender, where  $d_{bc}$  and  $D_{bc}$  are  $O(\kappa)$ , and security holds up to  $t$  corruptions and  $s$  slow parties, as long as  $2t + s < n$ , assuming a PKI setup for signatures. The theorem holds also unconditionally, if the setup consists of information-theoretic signatures [44].

► **Theorem 4.** *Let  $n, s, t$  be natural numbers such that  $2t + s < n$ . Assuming a PKI setup for signatures, there is an  $n$ -party asymmetric broadcast protocol with  $d_{bc} = D_{bc} = O(\kappa)$ , tolerating  $t$  malicious corruptions and  $s$  slow parties.*

Perhaps surprisingly, this is the best trade-off one can achieve, and tolerating  $2t + s = n$  for any non-trivial parameters  $t > 0$  and  $s > 0$  is impossible, even with setup, and for any number of fast and slow rounds.

► **Theorem 5.** *Let  $n, s, t > 0$  be natural positive numbers such that  $2t + s = n$ . Then, asymmetric broadcast is impossible against  $t$  malicious corruptions and  $s$  slow parties, even with setup.*

*A Protocol for  $t + s < n$  and  $t < n/2$ .* We now present an asymmetric MPC protocol where parties have access to asymmetric broadcast channels with fast and slow delays  $d_{bc}$  and  $D_{bc}$ , and achieve a protocol that is secure as long as  $t + s < n$  and  $t < n/2$ . The round complexity is  $O(n \cdot D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot n \cdot d_{bc})$  fast rounds. In the optimistic case where no party is corrupted, we save a factor  $n$  in the round complexity. That is, in this case the protocol incurs  $O(D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot d_{bc})$  fast rounds.

► **Theorem 6.** *Let  $n, s, t$  be natural numbers such that  $t + s < n$  and  $t < n/2$ . Assuming asymmetric broadcast with slow and fast delays  $D_{bc}$  and  $d_{bc}$ , there is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with guaranteed output delivery, in the presence of up to  $t$  corruptions and  $s$  slow parties. The round complexity is  $O(n \cdot D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot n \cdot d_{bc})$  fast rounds. In the optimistic case where no party is corrupted, the round complexity is  $O(D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot d_{bc})$  fast rounds.*

The number of rounds in the protocol above has a linear dependency in the number of parties in the worst case. This linear dependency can be removed at the cost of requiring any constant fraction of honest parties among the fast parties.

► **Corollary 7.** *Let  $\epsilon > 0$  and  $n, s, t$  be natural numbers such that  $t < \min\{(1 - \epsilon)(n - s), n/2\}$ . Assuming asymmetric broadcast with slow and fast delays  $D_{bc}$  and  $d_{bc}$ , there is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with guaranteed output delivery, in the presence of up to  $t$  corruptions and  $s$  slow parties. The round complexity is  $O(D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot d_{bc})$  fast rounds.*

## 2.2 MPC with Asymmetric Communication Cost

**The Model.** Similar to the previous model, we divide the parties into two categories: fast and slow parties. Parties have access to a complete network of standard synchronous point-to-point channels with the same delay upper bound. However, now the channels are differentiated with respect to the cost of communication. The channels between fast parties have a small cost and are denoted by *cheap channels*, and all other channels, i.e. the channels that contain at least one slow party as sender or receiver, have a high cost and are denoted by *expensive channels*.

We are interested in minimizing the number of bits transmitted over the expensive channels. Note that slow parties need to distribute their inputs, so the number of transmitted bits over expensive channels will definitely depend (at least) on the total number of slow parties and their input size. Our main focus is therefore that the number of bits over expensive channels does not depend on the circuit to evaluate.

**Existing Solutions.** Current solutions that make use of (multi-key) fully-homomorphic encryption (FHE) [40, 1, 26] already transmit a number of bits over expensive channels that is independent of the circuit to evaluate, given that the computation is performed under the homomorphic evaluation with no interaction.

**A Protocol From One-Way Functions.** Our focus is then on protocols that do not make use of FHE. Even more, we will focus on protocols that make use of as weaker cryptography assumptions as possible (preferably one-way functions, or no assumptions).

We provide a protocol that is resilient as long as  $t+s < n$  and  $t < n/2$  and achieves security with abort. The protocol communicates  $O(\text{poly}(n, \kappa))$  bits over the expensive channels and assumes the existence of one-way functions.

► **Theorem 8.** *Let  $n, s, t$  be natural numbers such that  $t + s < n$  and  $t < n/2$ . Assuming the existence of one-way functions, there is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with security with abort, in the presence of up to  $t$  malicious corruptions and  $s$  slow parties. The communication complexity is  $O(\text{poly}(n, \kappa))$  bits of expensive communication and  $O(\text{poly}(n, \kappa)|C|)$  bits of cheap communication.*

Interestingly, our communication-efficient protocol requires the existence of one-way functions, in contrast to our round-efficient protocols. We conjecture that this is necessary, and provide a partial result in this direction.

► **Lemma 9.** *Assume an  $n$ -party asymmetric MPC, secure up to  $t$  semi-honest corruptions and  $s$  slow parties such that the slow parties perform constant computation, for any  $n, s, t > 0$  natural numbers such that  $n - s > 1$ ,  $t + s < n$  and  $t < n/2$ . Then, this implies the existence of one-way functions.*

## 2.3 Open Questions

Our work initiates the area of asymmetric MPC and leaves several exciting new directions for future research. We highlight some of them:

1. General network topologies: We divide the parties into slow and fast parties, where fast parties have high-end channels among themselves, while all other channels are low-end. One can generalize this setting and consider more general network topologies, where a party may at the same time have some high-end and some low-end channels.
2. Other resource asymmetries: Our work considers models with respect to network asymmetries. One can explore other types of resources. For example, one can explore trade-offs with respect to asymmetric computation or memory resources.
3. Communication-efficient protocol with guaranteed output delivery: Our communication efficient protocols achieve security with abort. It would be interesting to see if our techniques extend to the case of guaranteed output delivery.
4. Concrete efficiency: Our protocols serve as a basis for feasibility of asymmetric MPC, and our focus is on minimizing the complexity coming from the slow parties. In practice, one may consider more refined settings where the cost difference is explicit (for example, expensive channels cost as twice as cheap channels). Understanding which concrete cost differences are relevant in practice and designing protocols tailored to such settings is an exciting open research direction.



### 3 Technical Overview

In this section we describe the techniques used to achieve our theorem statements.

#### 3.1 Information-Theoretic MPC with Asymmetric Delays

We present several information-theoretic solutions for the settings of security with abort and guaranteed output delivery.

**Security with Abort.** We provide a simple solution that achieves security with abort in a constant number of slow rounds, when  $t + s < n$  and  $t < n/2$ . The protocol lets all parties generate during a pre-processing phase OT correlations among the fast parties, using an information-theoretic MPC protocol for honest majority and security with abort, with round complexity linear in the depth of the circuit (e.g. [46, 21]). Since the OTs can be generated in parallel and can be computed with a constant-depth circuit, this takes a constant number of slow rounds. Then, the parties execute an information-theoretic MPC protocol in the client-server model, that achieves security with abort against a dishonest majority in the OT-hybrid model: The parties (acting as clients) distribute their inputs to the fast parties (acting as servers), who will compute the corresponding outputs and send them back to the respective parties. Existing protocols [36, 15] run in a constant number of rounds during the input and output phases, and the number of rounds during the computation is linear in the depth of the circuit to evaluate. This leads to Theorem 1.

We then show that our protocol achieves the optimal resilience: any asymmetric MPC protocol secure when  $t + s = n$  among  $n$  parties and with constant number of slow rounds, implies a synchronous MPC protocol with security with abort, resilient up to 1 corruption in constant number of rounds, which is a major open problem in round complexity of MPC. The proof implication follows from a simple emulation argument: In order to design a synchronous MPC protocol among  $s + 1$  parties, we simply let  $s$  parties emulate each of the slow parties in the asymmetric MPC protocol, and the last party to emulate all fast parties altogether (in total  $t$  parties). The resulting synchronous protocol has a round-complexity that is the same as the number of slow rounds in the asymmetric MPC protocol, and therefore the implication follows. This corresponds to Theorem 2.

**Asymmetric Broadcast.** We briefly sketch the arguments that exactly characterize the feasibility of asymmetric broadcast.

First observe that it is clear that for a sender that is a slow party, it is impossible to expect the fast parties to obtain output fast, even when all parties are honest. Therefore, we focus on the case where the sender is a fast party.

The protocol to achieve asymmetric broadcast (for a fast sender), for any  $2t + s < n$ , assuming a PKI for signatures, is quite simple: Fast parties run a synchronous broadcast protocol [33] among themselves, and reach agreement on a value  $v$ . All fast parties send the value  $v$  to the slow parties, who take a majority decision. Since there is honest majority among the fast parties, all slow parties output the same value. This is presented in Theorem 4.

In order to show that asymmetric broadcast (for a fast sender) is impossible when  $2t + s = n$ , even with setup, we make use of two ideas. First, observe that fast parties must output before the slow parties are even able to communicate with any fast party. This is because fast rounds may be much faster than slow rounds, i.e., the delay  $\delta \ll \Delta$  of fast channels could be much smaller than the delay  $\Delta$  of slow channels, and asymmetric broadcast requires fast parties to output fast. Let  $v$  denote the value that the fast parties output.

Second, since there is a dishonest majority among the fast parties, the corrupted parties (including the sender) can simulate towards the slow parties an execution with input value  $v' \neq v$ . As a consequence, the honest slow parties cannot decide on a consistent output value. A precise scenario-based proof is presented in Theorem 5.

**Guaranteed Output Delivery.** We present two results. We first present a somewhat simple solution for an information-theoretic protocol in the regime where  $2t + s < n$  and  $t < n/3$ . The protocol works as follows: First, since  $t < n/3$ , the protocol generates a setup for information-theoretic signatures to emulate synchronous broadcast channels with guaranteed termination [44, 33] from slow parties to all parties in  $O(\kappa)$  slow rounds, and from fast parties to themselves in  $O(\kappa)$  fast rounds, where  $\kappa$  is the security parameter. Using these broadcast channels, parties can execute an existing synchronous protocol in the client-server model as follows: All parties initially play the role of a client, while each fast party in addition plays the role of a server. The clients distribute their inputs towards the  $n - s$  servers, where each synchronous round corresponds to a slow round. The servers then perform the protocol computation, where each synchronous round corresponds to a fast round. Finally, the fast parties robustly reconstruct each output to the respective clients, where each synchronous round corresponds to a slow round. Standard information-theoretic protocols [46, 14] tolerate up to half of the corrupted servers (we assume  $2t < n - s$ ) and any number of clients, and have a constant number of rounds and broadcast invocations, during the input and the output phase, and a number of rounds proportional to the circuit depth during the computation phase. This results in an asymmetric MPC protocol with  $O(\kappa)$  slow rounds and a number of fast rounds proportional to the circuit depth times  $\kappa$ , corresponding to Theorem 3.

We then present our information-theoretic asymmetric MPC protocol with guaranteed output delivery, and resilience  $t + s < n$  and  $t < n/2$ , which assumes asymmetric broadcast.

The protocol follows the sharing-based paradigm, and has a preprocessing phase and an online phase. During the preprocessing phase, the parties generate raw data that is independent of the inputs. During the online phase, the parties receive their inputs and perform the protocol evaluation.

In the preprocessing phase, we generate *certified* Beaver multiplication triples, using the MPC protocol by Cramer et. al. [14].

*Background.* Let us first recap their VSS protocol  $\Pi_{\text{vss}}$  [14]. The protocol follows traditional verifiable secret sharing schemes [19, 7] with bivariate polynomials, but uses so-called *information-checking* (IC) signatures, instead of error correction. One can think about such signatures as information-theoretic signatures that can only be forwarded once. These can be generated unconditionally without setup, and also have a *linearity* property, where given signatures for values  $x$  and  $y$ , one can compute a signature on  $x + y$ .

In order to share a value  $v$ , the dealer  $D$  creates a random bivariate polynomial  $f(x, y)$  of degree at most  $t$ , with  $f(0, 0) = v$ . The univariate polynomial projections  $f(x, i)$  and  $f(i, y)$  are sent to party  $P_i$  in a signed manner (by sending all the points  $(a_{i1}, \dots, a_{in}) = (f(i, 1), \dots, f(i, n))$  and  $(b_{1i}, \dots, b_{ni}) = (f(1, i), \dots, f(n, i))$ , where each point is signed using IC-signatures). After this, the parties can bilaterally compare the cross-point values between them, and expose inconsistent behavior by the dealer by broadcasting the signatures. If an inconsistency is detected, the dealer is disqualified.

After the checking process, the values held by honest parties are consistent, and since there are at least  $n - t \geq t + 1$  honest parties, these values uniquely define a bivariate polynomial  $f'(x, y)$  of degree at most  $t$ , which in turn defines a fixed secret  $v'$  (which is  $v' = v$  if the dealer is honest). Therefore, this already ensures that the dealer is committed to a value after the sharing phase.



Still, the reconstruction might fail if the adversary sends corrupted shares (the adversary can send arbitrary shares). To avoid that, each share of  $P_i$  is also signed by the other parties. This will in turn prevent the adversary from corrupting the secret at reconstruction time.

At the end of the VSS, each party  $P_i$  holds sub-shares  $(a_{i1}, \dots, a_{in})$ , where  $a_{ij}$  is signed by  $P_j$ . This implicitly defines a share  $a_i$ , which in the case of an honest dealer is  $f(i, 0)$ .

With the above VSS, one can process addition gates locally (using the fact that the IC-signatures are linear). The multiplication gates are processed using the well-known method by Gennaro, Rabin and Rabin [22]: Each party  $P_i$  locally multiplies his shares  $a_i$  and  $b_i$  of the input wires  $a$  and  $b$ , and shares the result  $d_i = a_i b_i$  using VSS. This results in  $n$  VSSs and a proper sharing of the output wire  $c$  can be computed as a fixed linear combination of these. The authors show a way for  $P_i$  to share a secret  $d_i$ , such that  $d_i = a_i b_i$  and to prove that he has done so properly. The details can be found in [14].

*The Online Protocol.* At the start of the online phase, enough triples  $(x, y, z)$  have been shared using the protocol described above, where each of the shares  $x_i, y_i, z_i$  are held (implicitly) by party  $P_i$  via the corresponding sub-shares, which are IC-signed by the other parties. Note that generating such certified Beaver triples takes  $O(1)$  invocations of broadcast, since they can be generated in parallel.

The online phase proceeds as follows. Parties distribute their inputs using  $\Pi_{\text{VSS}}$ . The addition gates are locally computed (simply adding the shares and the IC-signatures, since they are linear). In the multiplication gates, fast parties publicly open two random values,  $(a - x)$  and  $(b - y)$ , where  $a$  and  $b$  are the values of the input wires to the multiplication gate, by running the same reconstruction procedure of  $\Pi_{\text{VSS}}$ , except that they distribute their shares using asymmetric broadcast. Since the values are IC-signed, corrupted fast parties can only withhold their shares. Since fast parties distribute their shares via the asymmetric broadcast channel, this implies that all parties, fast and slow, reach agreement on the set of parties that did not contribute their share and are corrupted.

Note that since the threshold is  $t < n - s$ , if all fast parties contribute their shares, the slow parties do not need to participate (and the protocol can proceed between between the fast parties, without incurring additional slow rounds). However, if not all shares are received, a process to identify and kick out corrupted parties is performed: fast parties wait for the slow parties to help opening the shares (note that  $n - t > t$ , and therefore all honest parties can jointly open the shares). The corrupted identified parties are then kicked out of the computation, and the protocol is restarted without the kicked parties. This process incurs an overhead of a constant number of slow asymmetric broadcast delays. And since every time at least one corrupted party is kicked out, the incurred overhead on the total number of slow rounds is linear in the number of parties. This corresponds to Theorem 6.

### 3.2 MPC with Asymmetric Communication Cost

We describe the protocol for MPC that communicates  $O(\text{poly}(n, \kappa))$  bits over slow connections, and achieves resilience  $t + s < n$  and  $t < n/2$ . The protocol is based on one-way functions, and is similar to the simple protocol mentioned in Theorem 1 in the asymmetric delay model.

In that protocol, the step that is communication expensive, is the generation of the OT correlations, which depends on the circuit size. In order to solve that, we will make use of OT-extension protocols [41, 2], which can be based on one-way functions. More concretely, since  $t < n/2$ , parties can jointly create  $\kappa$  OT correlations among each pair of fast parties using an honest-majority MPC protocol [46, 14]. This step communicates  $O(\text{poly}(n, \kappa))$  bits over slow connections. The fast parties then perform an OT-extension protocol to set up

## 6:10 Asymmetric Multi-Party Computation

an OT channel between each pair of fast parties [41, 2]. With this setup, parties can then perform an unconditional protocol achieving dishonest majority in the OT-hybrid model [36, 15] among the fast parties. This is stated in Theorem 8.

The protocol described above makes use of one-way functions. We conjecture that this is necessary, and provide a partial result: we show that any asymmetric MPC protocol containing at least two fast parties, where the slow parties perform little computation, and with resilience  $t + s < n$ , implies the existence of one-way functions.

The high-level idea is to build an OT extension protocol from an asymmetric MPC protocol. Since OT extension implies the existence of one-way functions [38], the claim follows. Assume that there is an asymmetric MPC protocol that outputs a large number of OTs. We can emulate the computation of each slow party using a protocol for dishonest majority (e.g. [24]). Note that since each slow party performs a small amount of computation, the circuit that is used to emulate the computation uses a small number of (seed) OTs as well. This is stated in Lemma 9.

### 4 Models and Definitions

We consider a set of  $n$  parties  $\mathcal{P} = \{P_1, \dots, P_n\}$ . We partition the set of parties into two known categories, slow parties and fast parties,  $\mathcal{P} = \mathcal{S} \sqcup \mathcal{F}$ . Let  $\kappa$  be the security parameter.

#### 4.1 Communication Network and Adversary

We consider a complete network of point-to-point secure channels. Parties have access to synchronized clocks, and messages sent by honest parties are guaranteed to be delivered within some known upper bound delay. We consider two asymmetric network models.

**Network with Asymmetric Delays.** In the first model, we consider a network with asymmetric delays. The channels between fast parties deliver messages within a small delay  $\delta$ , and are denoted fast channels. And all channels containing at least one slow party have a large delay  $\Delta$ , and are denoted slow channels. We measure the round complexity as the number of slow P2P-rounds (communication steps via all channels), and the number of fast P2P-rounds (communication steps via fast channels). We will omit mentioning the P2P term, and simply denote such rounds as fast and slow rounds.

**Network with Asymmetric Communication Cost.** In the second model, we consider a network with asymmetric communication cost. Here, all the channels have the same delay upper bound, similar to the standard synchronous network model, but the cost of communication is asymmetric. We will consider expensive communication, the number of bits transmitted via channels that contain at least one slow party, and cheap communication, the number of bits transmitted via channels that contain only fast parties.

**Adversary.** We consider a static adversary who corrupts parties in an arbitrary manner at the beginning of the protocol.

#### 4.2 Broadcast

Broadcast allows a designated party called the *sender* to consistently distribute a message among a set of receivers.

**Synchronous Broadcast.** The synchronous broadcast channel with guaranteed termination delivers the output to the set of receivers after a fixed number of rounds.<sup>1</sup> Synchronous broadcast protocols with guaranteed termination can be achieved within  $O(\kappa)$  rounds, when there are up to a third fraction of corrupted parties [19]. This is also the case for honest majority, if a setup is available [33]. In the dishonest majority setting, synchronous broadcast is achievable in  $O(n)$  rounds with a PKI setup [18], and even unconditionally with a setup for information-theoretic signatures [44].

These protocols, when run in the asymmetric network delay model, achieve an actual number of rounds that is proportional to the slowest channel. This means, that if all the parties involved (sender and receivers) are connected via fast channels, the output is received after a fixed number of fast rounds. However, when some of the channels between the considered parties are slow, the protocols guarantee that the receivers obtain the output in a fixed number of slow rounds.

### 4.3 Secret Sharing

In some of our protocols, we make use of Shamir secret sharing scheme [48]. This is a  $t$ -out-of- $n$  linear secret-sharing scheme over a finite field  $\mathcal{F}$ , consisting of two protocols, (Sh, Rec), called share and reconstruct.

Protocol **Sh** allows a designated party, called the dealer, to distribute a value  $s \in \mathcal{F}$  among  $n$  parties,  $P_1, \dots, P_n$ . For that, the dealer samples a uniform random polynomial  $f \in \mathcal{F}[x]$  with degree at most  $t$ , and subject to the fact that  $f(0) = s$ . Then, the dealer sends the value  $f(i) = s_i$  to  $P_i$ . We denote  $s_i$  the share of  $P_i$ , and the vector  $[s]_t = (s_1, \dots, s_n)$  is called a degree- $t$  sharing of  $s$ . We may omit the degree if it is clear from the context. Note that any set of  $t$  shares does not reveal anything about the secret. Protocol **Rec** allows parties to jointly reconstruct a secret  $s'$ , which corresponds to the original secret  $s$  if the dealer is honest. Shamir secret sharing scheme satisfies in addition the following properties:

- Additive Homomorphism:  $\forall [x]_t, [y]_t, [x + y]_t = [x]_t + [y]_t$ .
- Local Multiplication of Degree- $t$  Sharings:  $\forall [x]_t, [y]_t, [x \cdot y]_{2t} = [x]_t \cdot [y]_t$ .

### 4.4 Oblivious Transfer

Oblivious transfer [45] is a two-party primitive between a sender  $S$ , and a receiver  $R$ . The sender has two inputs  $x_0, x_1 \in \{0, 1\}$ , called the messages, and the receiver  $R$  has an input  $c \in \{0, 1\}$ , called the selection bit. The oblivious transfer guarantees that  $R$  outputs  $x_c = c(x_0 \oplus x_1) \oplus x_0$ , and that no party learns any other information.

## 5 MPC with Asymmetric Delays

In this section we introduce protocols in the model with asymmetric delays. We are interested in protocols that incur as few slow rounds as possible, preferably a constant, and tolerating a high number of corruptions and slow parties.

<sup>1</sup> There are also protocols with probabilistic termination [19, 33], where the parties obtain output after an *expected-constant* number of rounds. However, composing such protocols involves many subtleties. See for example [13] for a nice discussion.

## 5.1 Security with Abort

**Protocol Description.** We present a naive protocol that achieves security with abort in a constant number of slow rounds, when  $t + s < n$  and  $t < n/2$ .

The protocol lets all parties generate during a pre-processing phase OT correlations among the fast parties, using a (synchronous) information-theoretic MPC protocol for honest majority and security with abort (e.g. [46, 21]), with round complexity linear in the circuit depth to evaluate. Note that since the OTs can be generated in parallel and can be computed with a constant-depth circuit, this is possible in a constant number of slow rounds. Then, the parties execute an information-theoretic MPC protocol in the client-server model, that achieves security with abort against a dishonest majority in the OT-hybrid model: All parties act as clients and distribute their inputs to the fast parties, who also act as servers. The fast parties will then compute the corresponding outputs and send them back to the respective parties. Existing protocols [36, 15] run in a constant number of rounds during the input and output phases, and the number of rounds during the computation is linear in the depth of the circuit to evaluate. As a consequence, the overall protocol incurs a constant number of slow rounds, and a number of fast rounds proportional to the depth of the circuit. This leads to the following theorem, and the proof follows from the security of [46, 21] and [15].

► **Theorem 1.** *Let  $n, s, t$  be natural numbers such that  $t + s < n$  and  $t < n/2$ . There is an information-theoretic asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with security with abort, in the presence of up to  $t$  malicious corruptions and  $s$  slow parties. The round complexity is  $O(1)$  slow rounds and  $O(\text{depth}(C))$  fast rounds.*

**Barrier Result.** Our result shows that improving the resiliency achieved by the protocols in the above sections would be a breakthrough in the area of information-theoretic synchronous MPC. In particular, if there is an information-theoretic protocol that is constant in the number of slow rounds for  $t + s = n$ , this implies a constant-round information-theoretic synchronous MPC protocol secure up to 1 corruption.

► **Theorem 2.** *Let  $n, s, t > 0$  be natural numbers such that  $t + s = n$ . Then  $n$ -party information-theoretic asymmetric MPC with security with abort (resp. guaranteed output delivery), resilient up to  $t$  corruptions and  $s$  slow parties in  $R$  slow rounds, implies  $(s + 1)$ -party information-theoretic synchronous MPC with security with abort (resp. guaranteed output delivery), resilient up to 1 corruptions in  $R$  rounds.*

**Proof.** Let  $\Pi$  be the protocol for fast and slow parties with  $R$  slow rounds. We want to construct a synchronous protocol  $\Pi'$  among  $s + 1$  parties with  $R$  rounds.

Let us call the  $s + 1$  virtual parties  $P_1, \dots, P_{s+1}$ . Each party  $P_i$ ,  $i \in [s]$  emulates a slow party, and the last party  $P_{s+1}$  emulates all the fast parties. The parties then execute the protocol  $\Pi$ , where the messages between fast parties are emulated internally by  $P_{s+1}$ .

The resulting protocol  $\Pi'$  is a secure synchronous protocol up to 1 corruption and with  $R$  rounds. This follows from the fact that  $\Pi$  is a secure protocol tolerating  $t$  corruptions, and each virtual party contains at most  $t$  parties from  $\Pi$ . ◀

## 5.2 Broadcast with Asymmetric Delays

The *asymmetric* broadcast channel guarantees the delivery of a consistent message fast to the fast parties and slow to the slow parties. More precisely, an asymmetric broadcast channel achieves guaranteed output after  $D_{bc}$  slow rounds for slow parties, and  $d_{bc}$  fast rounds for

the fast parties. We will make use of this channel in the protocol for guaranteed output delivery with resilience  $t + s < n$  and  $t < n/2$ , but in this section we study the feasibility of this primitive from a complete network of point-to-point channels as a stand-alone question.

**Functionality  $\mathcal{F}_{\text{sBC}}$**

- 1: On input  $x$  from the sender  $P^*$ , output  $x$  to the adversary. Then, output  $x$  to all parties in  $\mathcal{S}$  after  $D_{bc}$  slow rounds and all parties in  $\mathcal{F}$  after  $d_{bc}$  fast rounds.

First, note that when the sender is slow, it is impossible to achieve asymmetric broadcast, since one needs at least a slow round to distribute the value towards the fast parties. Therefore, in the following we focus on the more interesting case where the sender is a fast party.

**Feasibility.** Assuming setup, if  $2t + s < n$ , it is easy to see that asymmetric broadcast with a fast sender is achievable. The protocol proceeds as follows: The sender with input  $s$  uses a synchronous broadcast protocol to distribute his value among all the fast parties. Since there is an honest majority of fast parties, one can for example use the protocol by Katz and Koo [33]. All the fast parties reach agreement on a value  $s'$  (which is  $s$  if the sender is honest) within  $O(\kappa)$  fast rounds, and they send their value to the slow parties, who will take a majority decision and output the result. Therefore, the slow parties output after  $O(\kappa)$  fast rounds, and 1 slow round.

► **Theorem 4.** *Let  $\kappa$  be a security parameter. Further let  $n, s, t$  be natural numbers such that  $2t + s < n$ . Assuming a PKI setup for signatures, there is an  $n$ -party asymmetric broadcast protocol with  $d_{bc} = D_{bc} = O(\kappa)$ , tolerating  $t$  malicious corruptions and  $s$  slow parties.*

The protocol can be achieved with unconditional security, if a setup for information-theoretic signatures is assumed [44].

**Impossibility.** We show that asymmetric broadcast is impossible when  $2t + s = n$  for a fast sender (even with setup), for any non-trivial parameters  $t > 0$  and  $s > 0$ . Note that this is in contrast to synchronous broadcast, which is achievable for any number of corruptions assuming a PKI setup [18]. (Or even unconditionally, assuming information-theoretic signatures [44].) The main challenge is to achieve agreement between fast and slow parties. Intuitively, since asymmetric broadcast requires fast parties to obtain the output fast, they need to decide their output value (let us denote it  $v$ ) without having received any value from the slow parties. Moreover, since there is dishonest majority among the fast parties, they can act towards the slow parties as if their output value was  $v' \neq v$ . As a consequence, the honest slow parties will not output  $v$  and consistency is broken. The proof of the following theorem can be found in Appendix B.

► **Theorem 5.** *Let  $n, s, t > 0$  be natural positive numbers such that  $2t + s = n$ . Then, asymmetric broadcast is impossible against  $t$  malicious corruptions and  $s$  slow parties, even with setup.*

### 5.3 Guaranteed Output Delivery

In this section, we present two protocols. The first protocol achieves a lower resilience, but operates only assuming point-to-point channels. The second protocol has a higher resilience, but makes use of asymmetric broadcast.

**Protocol for  $2t + s < n$  and  $t < n/3$ .** In the regime where  $2t + s < n$  and  $t < n/3$ , it is easy to design an asymmetric MPC protocol, by simply delegating the computation to the fast parties. Note that since  $t < n/3$ , the parties can create a setup of information-theoretic signatures, which can be used to construct a synchronous broadcast channel [44, 33] from slow parties to all parties in  $O(\kappa)$  slow rounds, and from fast parties among themselves in  $O(\kappa)$  fast rounds. With the emulated synchronous broadcast channels, parties can then execute an honest majority protocol such as [46] as follows: the slow parties, using the emulated broadcast channel, use a verifiable secret sharing scheme to share their input towards the fast parties (with threshold  $t$ ), who will robustly evaluate the circuit among themselves. The fast parties can then robustly reconstruct the output towards the respective recipients. The total round complexity is  $O(\kappa)$  slow rounds and  $O(\text{depth}(C) \cdot \kappa)$  fast rounds. The proof of the following theorem follows from the security of [46] and [33].

► **Theorem 3.** *Let  $\kappa$  be a security parameter. Let  $n, s, t$  be natural numbers such that  $2t + s < n$  and  $t < n/3$ . There is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with guaranteed output delivery in the presence of up to  $t$  malicious corruptions and  $s$  slow parties. The round complexity is  $O(\kappa)$  slow rounds and  $O(\text{depth}(C) \cdot \kappa)$  fast rounds.*

We note that if a synchronous broadcast channel is given (or alternatively a setup for information-theoretic signatures), the condition  $t < n/3$  is not necessary.

**Protocol for  $t + s < n$  and  $t < n/2$ .** In this section, we present a protocol that achieves guaranteed output delivery with the higher trade-off  $t + s < n$  and  $t < n/2$ . The resulting protocol has round complexity  $O(n \cdot D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot n \cdot d_{bc})$  fast rounds, assuming asymmetric broadcast.

**Generating Certified Beaver Triples.** We generate Beaver multiplication triples  $(a, b, c)$ , that are shared among all parties. The triples are certified, in the sense that all shares are signed via *information-checking* signatures. There can be thought of as signatures that are information-theoretic, and can only be forwarded once. The signatures are also homomorphic, in the sense that for two values that can be verified by the scheme, any linear combination of them can also be verified with no additional information. See [14] for a concrete construction.

We generate the triples using the protocol by Cramer et al. [14], which makes use of such a IC-signature scheme. The protocol takes  $O(1)$  invocations to asymmetric broadcast (incurring a total of  $O(D_{bc})$  slow rounds), since the multiplication triples can be generated in parallel. In this protocol, a value  $s$  is shared using a bivariate polynomial  $f(x, y)$  with degree at most  $t$ . At the end of the sharing protocol, each honest party  $P_i$  holds the values  $s_{i1}, \dots, s_{in}$  that lie on a degree- $t$  polynomial, which in the case the dealer is honest, corresponds to the values  $f(i, 1), \dots, f(i, n)$ . This implicitly defines the share of  $P_i$ , which is  $s_i = f(i, 0)$ . Moreover, each value  $s_{ij}$  is signed by party  $P_j$ , and we denote such a signature (from  $P_j$  to  $P_i$ ) by  $\sigma(s_{ij}, P_j, P_i)$ . This signature allows  $P_i$  to forward the value  $s_{ij}$  in an authentic way. In Appendix C, we recap the protocol in detail.

**The Online Phase.** At the start of the online phase, enough triples  $(a, b, c)$  have been shared using the protocol in [14]. This means, that each party  $P_i$  implicitly holds each of the shares  $a_i, b_i, c_i$  via the corresponding sub-shares, which are signed by the other parties.

The online phase proceeds as follows. Parties distribute their inputs using  $\Pi_{\text{VSS}}$ , the VSS scheme in [14]. The addition gates can locally be computed (simply by locally adding the shares and locally adding the signatures, since they are linear). In the multiplication gates,



fast parties robustly open two random values,  $(x - a)$  and  $(y - b)$ , where  $x$  and  $y$  are the values of the input wires to the multiplication gate, by running the same reconstruction procedure of  $\Pi_{\text{vss}}$ , except that they distribute their shares using asymmetric broadcast. Since the values are signed, corrupted fast parties can only withhold their shares. Since fast parties distribute their shares via the asymmetric broadcast channel, this implies that all parties, fast and slow, reach agreement on the set of parties that did not contribute their share and are corrupted. Note that since the threshold is  $t < n - s$ , if all fast parties contribute their shares, the slow parties do not need to participate (and therefore we do not need to incur additional slow rounds). However, if not all shares are received, a process to identify and kick out corrupted parties is performed: fast parties wait for the slow parties to help opening the shares (note that  $n - t > t$ , and therefore all honest parties can jointly open the shares). The corrupted identified parties are then kicked out of the computation, and the protocol is restarted without the kicked parties. This process involves  $O(D_{bc})$  slow rounds. And since every time at least one corrupted party is kicked out, the incurred total number of slow rounds is linear in the number of parties times the broadcast slow delay. We formally describe the protocol in Appendix D, and a proof of the following theorem appears in Appendix E.

► **Theorem 6.** *Let  $n, s, t$  be natural numbers such that  $t + s < n$  and  $t < n/2$ . Assuming asymmetric broadcast with slow and fast delays  $D_{bc}$  and  $d_{bc}$ ,  $\Pi_{\text{rgod}}$  is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with guaranteed output delivery, in the presence of up to  $t$  corruptions and  $s$  slow parties. The round complexity is  $O(n \cdot D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot n \cdot d_{bc})$  fast rounds. In the optimistic case where no party is corrupted, the round complexity is  $O(D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot d_{bc})$  fast rounds.*

Although the number of slow rounds is independent of the depth of the circuit, it has the drawback that it depends on the number of parties in the worst case. However, if we assume that among the fast parties there is a constant fraction of parties that are honest, then we can modify the above protocol to achieve round complexity  $O(D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot d_{bc})$  fast rounds. Then, the adversary needs  $\epsilon(n - s)$  corrupted parties to not send messages in order to execute Steps 3 and 4 in a multiplication step, which will be all identified and kicked out of the computation. This can only happen at most  $1/\epsilon$  times.

► **Corollary 7.** *Let  $\epsilon > 0$  and  $n, s, t$  be natural numbers such that  $t < \min\{(1 - \epsilon)(n - s), n/2\}$ . Assuming asymmetric broadcast with slow and fast delays  $D_{bc}$  and  $d_{bc}$ , there is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with guaranteed output delivery, in the presence of up to  $t$  corruptions and  $s$  slow parties. The round complexity is  $O(D_{bc})$  slow rounds and  $O(\text{depth}(C) \cdot d_{bc})$  fast rounds.*

## 6 MPC with Asymmetric Communication Cost

In this section we introduce protocols in the model with asymmetric communication cost. We are interested in protocols that transmit as few bits as possible over expensive channels (channels containing at least one slow party), and that tolerate a high number of corrupted parties and slow parties. Looking closer, our protocol from Theorem 3 already achieves small expensive communication, independent of the circuit since expensive communication only occurs in the input and output stages. However, the resiliency is only  $2t + s < n$ . Therefore, we turn our attention to protocols achieving resiliency  $t + s < n$  and  $t < n/2$ .



## 6.1 Security with Abort

We provide a protocol that achieves security with abort and communicates  $O(\text{poly}(n, \kappa))$  bits over the expensive channels. The protocol assumes the existence of one-way functions, and follows from existing results. The idea is to generate pair-wise OT correlations among the fast parties during a pre-processing phase. This is possible [46, 21] since there is an honest majority  $t < n/2$  of parties, and the total number of communicated bits over expensive channels is independent of the circuit size. More concretely, all parties will prepare  $\kappa$  OT correlations per fast connection (in total  $O(n^2\kappa)$  OTs<sup>2</sup>). Each pair of fast parties can then use OT extension protocols [4, 32, 41] to set up OT channels between them. Once the OT channels are prepared, we can execute a standard unconditional OT-based protocols for dishonest majority [36, 15] to perform the computation among the fast parties, and deliver the outputs to all the parties, leading to the following theorem statement.

► **Theorem 8.** *Let  $n, s, t$  be natural numbers such that  $t + s < n$  and  $t < n/2$ . Assuming the existence of one-way functions, there is an asymmetric MPC protocol among  $n$  parties that securely evaluates circuit  $C$  with security with abort, in the presence of up to  $t$  malicious corruptions and  $s$  slow parties. The communication complexity is  $O(\text{poly}(n, \kappa))$  bits of expensive communication and  $O(\text{poly}(n, \kappa)|C|)$  bits of cheap communication.*

## 6.2 Barriers on Communication Complexity

In this section, we show that if one assumes an asymmetric MPC, where the slow parties perform a small amount of computation, then this implies OT extension.

► **Lemma 9.** *Assume an  $n$ -party asymmetric MPC, secure up to  $t$  semi-honest corruptions and  $s$  slow parties such that the slow parties perform constant amount of computation, for any  $n, s, t > 0$  natural numbers such that  $n - s > 1$ ,  $t + s < n$  and  $t < n/2$ . Then, this implies the existence of one-way functions.*

**Proof.** The proof strategy is to build a semi-honest OT extension protocol from a semi-honest asymmetric MPC protocol. Since OT extension for semi-honest corruption implies the existence of one-way functions [38], the claim follows. Let  $c$  the total circuit size to represent the computation of the slow parties (this has polynomial size). Assume that there is an  $n$ -party asymmetric MPC protocol  $\Pi$  with at least 2 fast parties and secure up to  $t = n - s - 1$  and  $t < n/2$ . Further assume that the protocol outputs  $c + 1$  OT correlations. We first describe an  $(n - s)$ -party reactive functionality  $\mathcal{F}_{\text{slow}}$  that emulates the computation of the slow parties. Concretely, this is a reactive functionality which keeps the internal joint state of the slow parties, and which, upon giving inputs from the fast parties, it updates its internal joint state and computes the outputs to the fast parties according to the protocol description  $\Pi$  for the slow parties. Now we create a two-party protocol  $\Pi'$  for OT extension, where both parties have access to the reactive functionality described above, and where:  $P_1$  emulates  $n - s - 1$  fast parties, and  $P_2$  emulates the 1 remaining fast party. The two parties execute the asymmetric MPC protocol, where any interaction with the slow parties is performed via the reactive functionality  $\mathcal{F}_{\text{slow}}$  instead. First, note that the reactive functionality can be realized using a dishonest majority synchronous MPC protocol such as GMW among

---

<sup>2</sup> This can be reduced to  $O(n\kappa)$  OTs if there is a constant fraction of honest fast parties, i.e.  $t < (1-\epsilon)(n-s)$ , for  $\epsilon > 0$  constant, using the results of Harnik, Ishai and Kushilevitz [30], which combines from distributing computations among several committees from Bracha [10], techniques for combining oblivious transfers from Harnik et al. [31], and constructions of dispersers [25, 47])

the fast parties. Further note that since we are assuming that the slow parties performs computation represented by a total circuit size  $c$ . This means that the number of OTs needed to realize  $\mathcal{F}_{\text{slow}}$  is  $c$ . Finally, in terms of security, note that since the asymmetric MPC protocol tolerates  $t = n - s - 1$  corruptions, the resulting two-party protocol tolerates 1 corruption. In conclusion, the resulting two-party protocol where the reactive functionality is emulated among the two fast parties is a secure OT extension protocol against a semi-honest static adversary, which implies one-way functions. This concludes the proof. ◀

---

## References

- 1 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4\_29.
- 2 Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 673–701. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5\_26.
- 3 Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: Laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 120–150. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4\_5.
- 4 Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996. doi:10.1145/237814.237996.
- 5 Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990. doi:10.1145/100216.100287.
- 6 Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure MPC with linear communication complexity. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 213–230. Springer, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8\_13.
- 7 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.
- 8 Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8\_17.
- 9 Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 353–380. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64375-1\_13.
- 10 Gabriel Bracha. An  $o(\log n)$  expected rounds randomized byzantine generals protocol. *Journal of the ACM (JACM)*, 34(4):910–920, 1987.
- 11 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.
- 12 Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid MPC: Secure multiparty computation with dynamic participants. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 94–123, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84245-1\_4.

- 13 Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3\_9.
- 14 Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 311–326. Springer, Heidelberg, May 1999. doi:10.1007/3-540-48910-X\_22.
- 15 Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 110–123. Springer, Heidelberg, August 1995. doi:10.1007/3-540-44750-4\_9.
- 16 Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 572–590. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5\_32.
- 17 Ivan Damgård, Daniel Escudero, and Antigoni Polychroniadou. Phoenix: Secure computation in an unstable network with dropouts and comebacks. Cryptology ePrint Archive, Report 2021/1376, 2021. URL: <https://ia.cr/2021/1376>.
- 18 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
- 19 Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988. doi:10.1145/62212.62225.
- 20 Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8\_16.
- 21 Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In David B. Shmoys, editor, *46th ACM STOC*, pages 495–504. ACM Press, May / June 2014. doi:10.1145/2591796.2591861.
- 22 Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In Brian A. Coan and Yehuda Afek, editors, *17th ACM PODC*, pages 101–111. ACM, June / July 1998. doi:10.1145/277697.277716.
- 23 Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84245-1\_3.
- 24 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420.
- 25 Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997.
- 26 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7\_4.
- 27 Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional MPC with guaranteed output delivery. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 85–114. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7\_4.

- 28 Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 618–646. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1\_22.
- 29 Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 499–529. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7\_18.
- 30 Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 284–302. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5\_16.
- 31 Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 96–113. Springer, Heidelberg, May 2005. doi:10.1007/11426639\_6.
- 32 Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4\_9.
- 33 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, August 2006. doi:10.1007/11818175\_27.
- 34 Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8\_21.
- 35 Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9\_36.
- 36 Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988. doi:10.1145/62212.62215.
- 37 Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8\_10.
- 38 Yehuda Lindell and Hila Zarosim. On the feasibility of extending oblivious transfer. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 519–538. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2\_29.
- 39 Chen-Da Liu-Zhang, Julian Loss, Ueli Maurer, Tal Moran, and Daniel Tschudi. MPC with synchronous security and asynchronous responsiveness. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 92–119. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4\_4.
- 40 Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5\_26.
- 41 Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5\_40.
- 42 Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th FOCS*, pages 404–415. IEEE Computer Society Press, October 2003. doi:10.1109/SFCS.2003.1238214.

- 43 Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 3–33. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8\_1.
- 44 Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 337–350. Springer, 1992.
- 45 Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. URL: <https://eprint.iacr.org/2005/187>.
- 46 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989. doi:10.1145/73007.73014.
- 47 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 3–13. IEEE, 2000.
- 48 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.
- 49 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25.

## Appendix

### A Related Work

To the best of our knowledge, all current protocols consider a *symmetric* network resource model, where all parties have access to the same types of channels: all channels have the same delay and the same communication cost. In contrast, we consider a more refined *asymmetric* resource model, by making a distinction between two party categories. In the following, we show how our results compare to previous existing results.

**Round Complexity of Information-Theoretic Synchronous MPC.** The round complexity of MPC has a significant line of works. We focus on the information-theoretic setting. Here, all known protocols incur a number of rounds, that is linear in the depth of the computed circuit [7, 11, 46], and each round corresponds to the worst-case delay upper bound  $\Delta$ . This is the case even when only one channel has delay  $\Delta$ , and all other channels have delay  $\delta \ll \Delta$ . In contrast, we provide protocols where the number of slow  $\Delta$ -rounds is independent of the circuit size (even constant in some cases). Note that a constant number of slow rounds is necessary in order to distribute the inputs of all slow parties.

**Responsive MPC.** A line of works focused on the problem of obtaining MPC protocols that are *responsive*. These are protocols where the running time is as fast as the actual network delay. Note that any asynchronous protocol is responsive, but there are also synchronous protocols that remain responsive when the number of corruptions is small [43, 39]. These protocols inherently cannot achieve input completeness, i.e., the inputs of up to  $t$  honest parties may be ignored from the computation, and are responsive up to  $t < n/3$  corruptions (above  $n/3$  corruptions, the running time is similar to that of synchronous protocols). In contrast, our protocols achieve input completeness, and benefit from the fast channels even up to  $t < n/2$  corruptions.

**Communication Complexity of MPC.** The communication complexity of MPC has a significant line of works as well. Traditional solutions incur  $\text{poly}(n)|C|$  communication, and since then, there has been a significant progress in the communication efficiency (see e.g. [16, 6, 27, 28]). All these works consider the communication cost to be the same for all channels, and, as a consequence, their communication depends on the circuit size. Our work makes a distinction between cheap and expensive channels, and we provide protocols where the amount of bits transmitted over expensive channels is independent of the circuit size.

**MPC with Sporadic Participation.** A line of works [29, 3, 9, 12, 23, 17] considered MPC protocols where not all parties need to be online and/or participate at all steps of the protocol. In such protocols, a main challenge is to keep the state of the honest parties consistent throughout the protocol execution, and the protocol efficiency is counted uniformly (there is no asymmetry between network resources). In our case all parties are always online, and the main challenge is on minimizing the use of resources from a fixed known set of (slow) parties.

## **B** Proof of Theorem 5

We prove by contradiction. Assume that there is such a secure asymmetric broadcast protocol resilient up to  $t$  corruptions and  $s$  slow parties such that  $2t + s = n$ , for  $s, t > 0$  and  $n - s > 1$ . We partition the set of parties into three sets: the slow parties  $\mathcal{S}$ , and two sets of size  $t$   $\mathcal{F}_0$  and  $\mathcal{F}_1$  for the fast parties. We make the argument for an external fast sender role  $P^*$ .

The adversary corrupts the sender  $P^*$  and chooses a random bit  $b$ . Then, it corrupts  $\mathcal{F}_b$ . In addition, the adversary runs the following two executions  $\text{Exec}_0$  and  $\text{Exec}_1$  at the same time.

- In  $\text{Exec}_b$ , the corrupted parties  $\mathcal{F}_b$  and the corrupted sender  $P^*$  run an execution with the other parties where the sender has input  $b$ , but where  $\mathcal{F}_b$  ignores every message from  $\mathcal{F}_{1-b}$  and does not communicate with them either.
- In  $\text{Exec}_{1-b}$ , the corrupted sender  $P^*$  runs an execution with the slow parties and parties in  $\mathcal{F}_{1-b}$ , where the sender has input  $1 - b$ , and the parties in  $\mathcal{F}_b$  do not send any message to  $\mathcal{F}_{1-b}$ .

Consider an execution with the above adversary strategy. First, note that no matter what value  $b$  is chosen, the view of the slow parties  $\mathcal{S}$  is exactly the same. This is because the view of the slow parties consist of two independent executions, where  $\mathcal{F}_b$  and  $\mathcal{F}_{1-b}$  do not communicate, but otherwise follow an execution where the sender has input  $b$  and  $1 - b$ , respectively.

Second, note that when the bit  $b$  is chosen, the parties in  $\mathcal{F}_{1-b}$  output  $1 - b$ . This is because the view of  $\mathcal{F}_{1-b}$  is identically distributed as in an execution where the sender is honest with input  $1 - b$ , and parties in  $\mathcal{F}_b$  crashed. Note that  $\mathcal{F}_{1-b}$  must output fast, before receiving any message from  $\mathcal{S}$ .

The above observations imply that the slow parties output a value that is inconsistent with  $\mathcal{F}_{1-b}$ , with probability  $1/2$ . Note that even though  $\mathcal{S}$  sees the sender is corrupted, he does not see whether  $\mathcal{F}_0$  or  $\mathcal{F}_1$  is honest (these sets output different bits).



**C Recap of Protocol [14]**

In this section, we describe the details of the protocol by Cramer et al. [14], which is used in the preprocessing phase of the protocol  $\Pi_{\text{rgod}}$  in Section 5.3.

**IC-Signatures.** Information checking (IC) [14] is a tool for authenticating data that is information-theoretic. We make use of information checking among  $n$  parties  $P_1, \dots, P_n$ . Protocol **Dist** will be carried out by the dealer  $D$  with intermediary  $INT$  and the receivers  $P_1, \dots, P_n$ , each with the same input value  $s$ . The information sent by  $D$  to  $INT$  will be called an IC-signature and we denote such a signature as  $\sigma(s, D, INT)$ . In order to verify a signature among  $n$  parties, the **AuthVal** protocol is executed bilaterally by  $INT$  and each party  $P_i$ . Then, in protocol **Reveal**,  $INT$  broadcasts  $s$  and the authentication information, and if  $t + 1$  parties accept  $s$  then we say that the signature has been confirmed. These signatures enable  $D$  to give  $INT$  a signature which only  $INT$  can use to convince the other parties about the authenticity of a value received from the dealer. Therefore, we can use these IC-signatures as signatures given specifically from  $D$  to  $INT$ , so that  $INT$  can prove authenticity of a received value to any party.

**Verifiable Secret Sharing.** We recall the definition of verifiable secret sharing (VSS).

► **Definition 10.** A  $t$ -secure VSS scheme for sharing a secret  $s \in \mathbb{F}$  is a pair  $(\text{Sh}, \text{Rec})$   $n$ -party protocols that satisfy the following properties, even in the presence of an adversary corrupting up to  $t$  parties:

- *Correctness:* Once all honest parties terminate protocol **Sh**, there exists a fixed value,  $s' \in \mathbb{F} \cup \perp$ , such that the following requirements hold:
  - If the dealer  $D$  is honest, then  $s' = s$ , and each honest party outputs  $s'$  in protocol **Rec**.
  - If the dealer is corrupted then each honest party outputs  $s'$  in protocol **Rec**.
- *Privacy:* If the dealer is honest and no honest party has yet started **Rec**, then the adversary has no information about the shared secret  $s$ .
- *Termination:* If the dealer  $D$  is honest then all honest parties terminate **Sh**, and if the honest parties invoke **Rec**, then each honest party eventually terminates **Rec**.

**Protocol Description.** The protocol is based on classical protocols [19, 7], but using IC-signatures, instead of error correction.

In order to share a value  $s$ , the dealer  $D$  will make use of a bivariate polynomial  $f(x, y)$  of degree at most  $t$ . The projections  $f(x, i)$  and  $f(i, y)$  will be sent to party  $P_i$  (where all the points are signed using IC-signatures). The parties can now bilaterally compare the cross-point values between them, and expose inconsistent behavior by the dealer using the signatures. This implies that the values held by honest parties are consistent, and since there are at least  $n - t > t + 1$  honest parties, these values uniquely define a bivariate polynomial  $f'(x, y)$  of degree at most  $t$ , which in turn defines the secret. Therefore, this already ensures that the dealer is committed to a value after the sharing phase.

However, the reconstruction might still fail if the adversary sends corrupted shares. In order to avoid that, each share of  $P_i$  is also signed by the other parties. This will in turn prevent the adversary from corrupting the secret at reconstruction time.



**Protocol  $\Pi_{\text{VSS}}$** 

**Share** Let  $s$  be the input for the dealer  $D$ .

- 1:  $D$  chooses a random bivariate polynomial  $f(x, y)$  of degree at most  $t$  in each variable, such that  $f(0, 0) = s$ . Let  $s_{ij} = f(i, j)$ . The dealer sends to party  $P_i$  the values  $a_{1i} = s_{1i}, \dots, a_{ni} = s_{ni}$  and  $b_{i1} = s_{i1}, \dots, b_{in} = s_{in}$ . For each value  $a_{ji}, b_{ij}$ ,  $D$  attaches IC-signatures  $\sigma(a_{ji}, D, P_i)$ , and  $\sigma(b_{ij}, D, P_i)$ .
- 2: Party  $P_i$  checks that the two sets  $a_{1i}, \dots, a_{ni}$  and  $b_{i1}, \dots, b_{in}$  are  $t$ -consistent. If the values are not  $t$ -consistent,  $P_i$  broadcasts these values with  $D$ 's signature on them. If a party hears a broadcast of inconsistent values with the dealer's signature then  $D$  is disqualified and execution is halted.
- 3:  $P_i$  sends  $a_{ji}$  and a signature which he generates on  $a_{ji}$ ,  $\sigma(a_{ji}, P_i, P_j)$  privately to  $P_j$ .
- 4: Party  $P_i$  compares the value  $a_{ij}$  which he received from  $P_j$  in the previous step to the values  $b_{ij}$  received from  $D$ . If there is an inconsistency,  $P_i$  broadcasts  $b_{ij}$  and  $\sigma(b_{ij}, D, P_i)$ .
- 5: Party  $P_i$  checks if  $P_j$  broadcasted a value  $b_{ji}$ ,  $\sigma(b_{ji}, D, P_j)$  which is different than the value  $a_{ji}$  which he holds. If such a broadcast exists then  $P_i$  broadcasts  $a_{ji}$  and  $\sigma(a_{ji}, D, P_i)$ .
- 6: If for an index pair  $(i, j)$  a party hears two broadcasts with signatures from the dealer on different values, then  $D$  is disqualified and execution is halted.

**Reconstruct**

- 1: Party  $P_i$  broadcasts the values  $b_{i1}, \dots, b_{in}$  with the signature for value  $b_{ij}$  which he received from party  $P_j$ .
- 2: Party  $P_i$  checks whether  $P_j$ 's shares broadcasted in the previous step are  $t$ -consistent and all the signatures are valid. If not then  $P_j$  is disqualified.
- 3: The values of all non-disqualified parties are taken and interpolated to compute the secret.

**Multi-Party Computation.** Using the VSS scheme  $\Pi_{\text{VSS}}$  from above, it is easy to come up with an MPC protocol. Addition gates are straightforward and parties can process them locally (using the fact that the IC-signatures are linear). Multiplication gates are processed using the well-known method by Gennaro, Rabin and Rabin [22]: Each party  $P_i$  locally multiplies his shares  $a_i$  and  $b_i$  of the input wires  $a$  and  $b$ , and shares the result  $d_i = a_i b_i$  using VSS. This results in  $n$  VSSs and a proper sharing of the output wire  $c$  can be computed as a fixed linear combination of these. The authors show a way for  $P_i$  to share a secret  $d_i$ , such that  $d_i = a_i b_i$  and to prove that he has done so properly. The details can be found in [14].

**D Description of Protocol  $\Pi_{\text{rGod}}$** **Protocol  $\Pi_{\text{rGod}}$** 

Initialize  $t' = t$ ,  $\mathcal{F}' = \mathcal{F}$ .

**Preprocessing Phase**

- 1: Parties use the protocol [14] to create:
  - $n_m$  random sharings of certified Beaver triples  $([a_k], [b_k], [c_k])$ , where  $n_m$  is the number of multiplication gates, as explained before. This means, that for each triple  $(a, b, c)$ , each honest party  $P_i$  implicitly holds his share  $a_i$  by holding sub-shares  $a_{i1}, \dots, a_{in}$ , where each sub-share is IC-signed by  $P_j$ . These sub-shares lie on a degree- $t$  polynomial; and the shares  $a_i$  also lie on a degree- $t$  polynomial  $f$  with  $f(0) = a$ . And similarly, for the values  $b$  and  $c = ab$ .

**Input Phase** Let  $x_j$  be the input from party  $P_j$ .

- 1:  $P_j$  uses the protocol  $\Pi_{\text{VSS}}$  (described in Section C) to share his input  $x_j$ .

**Addition Gates**

- 1: Fast parties locally add the shares using linearity of the sharing scheme, and compute the corresponding IC-signatures using its linearity property.

**Multiplication Gates** Let  $[x]$  and  $[y]$  be sharings of the inputs to the gate.

- 1: Fast parties in  $\mathcal{F}'$  use a multiplication triple  $([a], [b], [c])$  to publicly reconstruct the values  $x - a$  and  $y - b$  among the fast parties, using the reconstruction procedure of  $\Pi_{\text{vss}}$  (where only the fast parties start). That is, the fast parties use asymmetric broadcast to distribute their share towards all parties, and the corresponding IC-signatures.
- 2: After time  $d_{bc}$ , all the fast parties reach agreement on whether all the shares received have correct IC-signatures are  $t$ -consistent. If not, they keep waiting for a total of  $2D_{bc}$  time. Otherwise, execute Step 5.
- 3: After time  $D_{bc}$ , all slow parties reach agreement on whether the shares distributed by the fast parties were correct ( $t$ -consistent and with correct signatures). If not, slow parties participate in the reconstruction by broadcasting their shares and IC-signatures.
- 4: After time  $2D_{bc}$ , either there were enough shares to reconstruct at Step 2, or all honest parties (fast and slow) identify at least one corrupted fast party  $P_k \in \mathcal{F}'$  that did not contribute its share. In this case, parties kick out the identified corrupted party, and restart the protocol, now with threshold  $t' = t - 1$ , and set  $\mathcal{F}' = \mathcal{F} \setminus \{P_k\}$ .
- 5: Fast parties in  $\mathcal{F}'$  locally compute a share of the output to the gate as  $[z] = (x - a)[b] + (y - b)[a] + [c] + (x - a)(y - b)$  (via locally adding each of the sub-shares), and update the IC-signatures accordingly using the linearity property.

**Output Phase**

- 1: To reconstruct a sharing  $[x]$  towards  $P_j$ , parties robustly reconstruct the secret to the  $P_j$ , using the VSS reconstruction protocol in  $\Pi_{\text{vss}}$ .

**E Proof of Theorem 6**

We describe the simulator for the online phase of the protocol, as the preprocessing phase is executed using the protocol [14] and can also be simulated.

**Simulation of the Input Phase.** For each honest party  $P_i$ , the simulator emulates an execution of  $\Pi_{\text{vss}}$  with input 0 towards the adversary. This includes sending random projected univariate polynomials to the adversary, along with IC-signatures. Each time a complain is received by the adversary, the simulator responds by broadcasting the corresponding stored share and signature.

For each corrupted party  $P_i$ , the simulator emulates an honest execution of  $\Pi_{\text{vss}}$  on behalf of the honest parties: it receives univariate polynomials with signatures on behalf of the honest parties. It then checks that they are all of degree- $t$ , and if not, it broadcasts the values with  $P_i$ 's signatures. Finally, for any share and signature that are broadcasted, if the corresponding party holding the cross-point share is honest, it checks whether that received share is consistent, and if not, it broadcasts these values with  $P_i$ 's signature on them.

**Simulation of Addition Gates.** The addition gates require no simulation, since they constitute local computation only.

**Simulation of Multiplication Gates.** During each multiplication gate, the simulator emulates the reconstruction procedure of  $\Pi_{\text{vss}}$  from the fast parties. That is, it emulates an execution of asymmetric broadcast, and gives to the adversary the shares and signatures corresponding

to  $x - a$  and  $y - b$ , where  $x$  and  $y$  are the input wire values in the simulated execution, and  $(a, b, c)$  is a multiplication triple. (Note that since  $a$  and  $b$  are random, the reconstructed values are also random.)

**Simulation of the Output Stage.** During the output phase, the simulator receives the output  $y_i$  for  $P_i$ , and the shares of the corrupted parties. With these, the simulator can consistently reconstruct the shares of the honest parties and send them to the adversary.

**Analysis of the Simulation.** We first argue about correctness. First, the pre-processing phase is successful, and all parties hold correct shares of the multiplication triples with correct signatures (with high probability). Similarly, after the input phase, parties hold correct shares of their inputs and signatures. This correctness trivially propagates to the rest of the wires in the circuit, since all operations are linear.

Now let us argue that the protocol securely computes the function. During the input phase for an honest party  $P_i$ , the adversary receives the projections of univariate polynomials  $f(x, j)$  and  $f(j, y)$ , for up to  $t$  indices  $j$ . By properties of bivariate polynomials, this reveals no information about the secret, and is therefore identically distributed as the univariate polynomials that the simulator reveals. In the multiplication phase, as hinted above, since the multiplication triple  $(a, b, c)$  contains uniform random values  $a$  and  $b$ , the reconstructed values are also distributed uniformly at random, identically as the simulated execution. This is also the case for the output gates, where the simulator outputs honest shares and signatures that are consistent with the corresponding reconstructed output.