Server-Aided Two-Party Computation with Minimal Connectivity in the Simultaneous Corruption Model

Ignacio Cascudo[1], Ivan Damgård [1], Oriol Farràs[2], and Samuel Ranellucci[1]

[1]Aarhus University,
{ignacio,ivan,samuel}@cs.au.dk
[2]Universitat Rovira i Virgili, Tarragona,
oriol.farras@urv.cat

Abstract

We consider secure two-party computation in the client-server model. In our scenario, two adversaries operate *separately but simultaneously*, each of them corrupting one of the parties and a restricted subset of servers that they interact with. We model security in this setting via the local universal composability framework introduced by Canetti and Vald and show that information-theoretically secure two-party computation is possible if and only if there is always at least one server which remains uncorrupted. Moreover, in our protocols each of the servers only needs to communicate with the two clients, i.e. no messages are exchanged directly between servers. This communication pattern is minimal.

Keywords: two-party computation, simultaneous corruption, universal composability with local adversaries, oblivious transfer.

1 Introduction

This paper considers secure computation, where two parties (Alice and Bob) hold inputs and want to compute an agreed upon function on those inputs in such a way that the intended output is the only new information released.

More specifically, our goal is to implement secure two-party computation with information theoretic security. It is well-known that this is impossible without assuming either preprocessing, additional auxiliary functionalities, or help from additional parties. The reason why shooting for unconditional security is nevertheless interesting is that information theoretic methods are typically computationally much more efficient than the heavy public-key machinery one needs for the two-party case if no additional help is assumed. In our approach, we assume that the two parties get help from n servers. This is essentially the client-server model of MPC introduced in [DI05], with 2 clients and n servers.

We depart from earlier work in this model in two ways. First, we strive for minimal connectivity. Our goal is to have each server communicate only with the two clients (i.e., the servers do not talk to each other). This communication pattern is minimal since, in order to be of any use in letting Alice and Bob do secure computation, a server must communicate with both of them. The minimality of the communication pattern has obvious advantages: it is easier to implement a protocol where servers do not need to communicate or even be aware of each other; in fact, as we will see later, in some cases the amount of work made by each of the servers in our protocol can even be made independent from the number of servers; it is also easier to set up the required secure communication channels if the servers only need public keys for Alice and Bob; finally, when running the protocol, a server can proceed as soon as he gets the next messages from Alice and Bob, instead of waiting to hear from all other servers, which would be required if the protocol was synchronous and involved interaction between all servers.

The second point that sets our work apart is the way we model adversarial behaviour. The standard model assumes one adversary who may corrupt one of the clients and some number of servers, typically some constant fraction of them. If we want unconditional security, we need to assume an honest majority (or $\mathcal{Q}_2$ for general adversary structures [HM00]). In this paper, we ask whether we can tolerate more corruptions and still have unconditional security, if we assume *two adversaries*, $\mathbb{A}, \mathbb{B}$ that operate *separately, but simultaneously*. We will characterise the capabilities of these adversaries by two anti-monotone families of server subsets $\mathcal{A}, \mathcal{B}$, so-called adversary structures.

We think of these adversary structures as follows: we want to design protocols such that if you manage to corrupt Alice and a set of servers in $\mathcal{A}$, you have of course broken Alice's privacy, but you should not be able to break Bob's – and vice versa for $\mathcal{B}$.

More concretely, $\mathbb{A}$ is allowed to corrupt Alice and a set of servers $A \in \mathcal{A}$ (it may decide to not corrupt servers). Corruption may be semi-honest or malicious. Likewise we allow $\mathbb{B}$ to corrupt Bob and a set $B \in \mathcal{B}$. We call this the *double adversary model*. An obvious question now is how we should define security for a protocol in this model. To this end, one can first observe that if either $\mathbb{A}$ or $\mathbb{B}$ decide to not corrupt anyone, we are back in the standard single adversary model and security means what it usually does. But we also define meaningful security requirements in the more general case where both adversaries operate at the same time. Loosely speaking, we require that an adversary should learn as little information as possible and be unable to affect the result achieved by parties that he did not corrupt. We will discuss this in more detail in a moment. Our main result gives a positive answer to the above question: it says that secure computation is possible in our model even in cases where all but one of the servers has been corrupted by one of the two adversaries, and moreover, this is the best we can hope for.

THEOREM 1.1 (INFORMAL) *The pair $(\mathcal{A}, \mathcal{B})$ is said to be $\mathcal{R}_2$ if for any $A \in \mathcal{A}, B \in \mathcal{B}$, $A \cup B$ is not the entire set of servers. Two-party computation is possible in the double adversary model w.r.t. adversary structures $\mathcal{A}$ and $\mathcal{B}$ if and only if $(\mathcal{A}, \mathcal{B})$ is $\mathcal{R}_2$.*

The $\mathcal{R}_2$ property looks a bit like the well known $\mathcal{Q}_2$ property for a single adversary structure, introduced by Hirt and Maurer [HM00][1]. $\mathcal{Q}_2$ is necessary and sufficient for statistically secure MPC in the single adversary case assuming also a broadcast channel.

For the case of threshold adversary structures, where we assume that Alice can corrupt at most t_A servers, and Bob can corrupt at most t_B, $\mathcal{R}_2$ means that $t_A + t_B < n$. However, more general cases may occur as well: assume, for instance, that there is a cost associated with corrupting each server, which does not have to be the same for all servers. Now, if both adversaries have a certain budget they can spend, this corresponds to certain adversary structures containing the sets that they can corrupt. In this case the $\mathcal{R}_2$ condition intuitively says that the joint wealth of the adversaries is limited to some extent. Interestingly, we will see that there are even examples of $\mathcal{R}_2$ pairs $(\mathcal{A}, \mathcal{B})$, yet neither $\mathcal{A}$ nor $\mathcal{B}$ is $\mathcal{Q}_2$. On the other hand, a pair $(\mathcal{A}, \mathcal{B})$ naturally defines an adversary structure $\mathcal{U}$ in the *extended set* P of $n + 2$ players consisting of Alice, Bob and the n servers, where the maximal sets in $\mathcal{U}$ are the sets that consist of Alice and a set from $\mathcal{A}$, or Bob and a set from $\mathcal{B}$. It turns out that $(\mathcal{A}, \mathcal{B})$ is $\mathcal{R}_2$ if and only if $\mathcal{U}$ is $\mathcal{Q}_2$.

This last observation means that Alice and Bob could also use general results on MPC for $\mathcal{Q}_2$ structures to compute their desired function securely. We want to emphasise that our result distinguishes itself from this approach in several respects: first, we already mentioned that general secure MPC protocols against an *active* adversary with a $\mathcal{Q}_2$ structure must assume the existence of a broadcast channel and can only achieve statistical security. In contrast, our protocol does not assume broadcast and achieves perfect security. Second, even if one makes the necessary additional assumptions, using generic MPC protocols would not achieve our minimal communication pattern, where each server only needs to talk to Alice and Bob and does not even need to be aware of the other servers. Finally, we are not aware of any results that imply that known generic MPC protocols are still secure in our double adversary model (see below for a more detailed description of this security notion).

We can already now observe that one part of the theorem is easy to show, namely if $(\mathcal{A}, \mathcal{B})$ is not $\mathcal{R}_2$ then we cannot hope for an unconditionally secure protocol. This is simply because a secure protocol for a non-$\mathcal{R}_2$ case would imply general 2-party unconditionally secure computation in the standard model which is well known to be impossible. This follows by a standard emulation argument: Consider semi-honest corruption and assume we have a multiparty protocol secure against corruption of Alice as well as server set A by $\mathbb{A}$ and Bob as well as server set B by $\mathbb{B}$, where $A \cup B$ is the full set of servers. Then the entire protocol can be emulated by two players where one plays for Alice and servers in A while another plays for Bob and servers in B. This would give us a secure two-party protocol in the standard model.

Techniques and Details

We now discuss our security notion. As we said, the corruption (if any) done by $\mathbb{A}, \mathbb{B}$ may be semi-honest or malicious, but not all combinations make sense.

The first case is when $\mathbb{B}$ does semi-honest corruption. This means that Bob will follow the protocol. Then we require that Bob gets correct output, no matter what $\mathbb{A}$ does. Furthermore, we require that $\mathbb{A}$ learns a minimal amount of information, which means that $\mathbb{A}$ learns (of course) Alice's private input, but nothing more, even under malicious corruption.

The second case is when $\mathbb{B}$ is malicious. Note that we do not put restrictions on the behaviour of malicious parties, and hence $\mathbb{B}$ could choose to send out his entire view to all players. This means that $\mathbb{A}$ would see information from a set in $\mathcal{A}$ and Alice's view as well, in addition to information from its own corruptions. Therefore, *in general*, we cannot hope to keep anything private from $\mathbb{A}$ in such a case. But a special case where we can give a guarantee is when $\mathbb{A}$ corrupts no one, since then it will of course learn nothing more than $\mathbb{B}$. Conditions symmetric to the above two are required when $\mathbb{A}$ is semi-honest, respectively malicious.

We will not treat the case where both adversaries are malicious: since they can choose to broadcast everything they know, it does not make sense to consider this as two separate adversaries.

[1] $\mathcal{A}$ is $\mathcal{Q}_2$ if for all $A, B \in \mathcal{A}$, $A \cup B$ is not the set of all players.

We also do not explicitly treat the case where adversaries corrupt servers in $\mathcal{A}$ or $\mathcal{B}$ only (and not Alice or Bob). In fact our model excludes this case. This is because we think of our problem as being inherently a two-party computation to be applied in scenarios where Alice believes that any attack against her will be initiated by Bob (and vice versa). Therefore we insist that if you corrupt a set of servers in $\mathcal{A}$, this counts as if you also corrupted Alice. This distinction may seem unnecessary in that an adversary who corrupts only servers seems weaker than one who may also corrupt Alice or Bob. This is true for semi-honest corruption, but the malicious case is different: with malicious servers only, we need to guarantee that the correct result is output, based on the honest inputs of Alice and Bob (that you cannot change in a simulation proof).

Nevertheless, our protocol does give meaningful (but non-standard) security gurantees for corrupted sets of only servers. We explain these in Appendix A where we also use an impossibility result from [FHM99] to argue that standard security cannot in general be obtained unless we put stronger conditions on $(\mathcal{A}, \mathcal{B})$ (and extend the security notion).

A technical detail we need to handle is that it may be the case that both adversaries corrupt the same server. In that case we just assume that both players see the entire view of the server in question. So, in particular, if $\mathbb{A}$ is semi-honest and $\mathbb{B}$ is malicious, $\mathbb{A}$ can see the potentially malicious behaviour of $\mathbb{B}$ in that server.

A few works have considered adversaries that operate separately and simultaneously. The closest to our scenario is [KMR11], where several clients are assisted in a (computationally) secure MPC protocol by a single server. They develop special-purpose security definitions for this problem.

Nevertheless, we choose to formalise the above ideas using the notion of Universal Composability with local adversaries of Canetti and Vald [CV12]. This allows us not only to consider separate adversaries but also to give guarantees for composition. On the other hand, some technical issues arise with respect to how we define the ideal functionality we implement, see more details within.

For the case of semi-honest adversaries, a protocol can be obtain in an almost straightforward manner from the results by Hirt and Maurer [HM00] on MPC secure against semi-honest adversaries, as we explain in Section 5. However, we also show a simpler protocol in that section whose complexity (in n) is essentially that of a linear secret sharing scheme realizing the structure $\mathcal{A}$.

Our protocol for the malicious case, described in Section 6, is considerably more involved. The difficulty comes from the fact that none of the adversary structures may be $\mathcal{Q}_2$, as mentioned above, and this means that we cannot directly use known solutions for verifiable secret sharing to commit the players to values they hold. Therefore, we cannot make our semi-honest solution be secure in the standard way. Instead, we use specially engineered linear secret sharings schemes for the two adversary structures to construct a protocol for oblivious transfer (OT) [Rab81, EGL82], and use the fact that OT is complete for two-party computation [Kil88]. The complexity (in n) of this protocol is again essentially that of the linear secret sharing scheme realizing $\mathcal{A}$. Quite surprisingly, the description of both our protocols does not involve the structure $\mathcal{B}$. The idea for our OT protocol is to let each of the servers implement an OT, which may be faulty because of the corruptions, and we suggest a new technique to make a secure OT from these n imperfect ones. In this regard, our notion is similar to that of OT combiners [HKN$^+$05, HIKN08]. However, the adversarial model we consider is different.

2 $\mathcal{Q}_2$ structures and $\mathcal{R}_2$ pairs of structures

We denote by $\mathcal{P}_n$ the set $\{1, 2, \ldots, n\}$. Furthermore, $2^{\mathcal{P}_n}$ is the family of all subsets of $\mathcal{P}_n$. An anti-monotone (or adversary) structure $\mathcal{A} \subseteq 2^{\mathcal{P}_n}$ is a family of subsets of $\mathcal{P}_n$ such that $\emptyset \in \mathcal{A}$ and for every $A \in \mathcal{A}$ and $B \subseteq A$ we have $B \in \mathcal{A}$.

DEFINITION 2.1 *We say that an adversary structure $\mathcal{A}$ is $\mathcal{Q}_2$ if for all $A, B \in \mathcal{A}$, we have $A \cup B \neq \mathcal{P}_n$.*

DEFINITION 2.2 *We say that a pair $(\mathcal{A}, \mathcal{B})$ of adversary structures is $\mathcal{R}_2$ if for all $A \in \mathcal{A}$, $B \in \mathcal{B}$, we have $A \cup B \neq \mathcal{P}_n$.*

$\mathcal{R}_2$ is a generalization of $\mathcal{Q}_2$. More precisely, the pair of adversary structures $(\mathcal{A}, \mathcal{A})$ is $\mathcal{R}_2$ if and only if $\mathcal{A}$ is $\mathcal{Q}_2$. However, there exist adversary structures $\mathcal{A}, \mathcal{B}$ such that neither $\mathcal{A}$ nor $\mathcal{B}$ are $\mathcal{Q}_2$, while the pair $(\mathcal{A}, \mathcal{B})$ is $\mathcal{R}_2$. For example: $n = 4$, and $\mathcal{A}$ and $\mathcal{B}$ are the adversary structures with maximal sets $\{1, 2\}, \{3, 4\}$ in the case of $\mathcal{A}$, and $\{1, 3\}, \{2, 4\}$ in the case of $\mathcal{B}$.

For an adversary structure $\mathcal{A}$, the dual adversary structure $\bar{\mathcal{A}}$ is defined as follows: $A \in \bar{\mathcal{A}}$ if and only if $\bar{A} \notin \mathcal{A}$, where $\bar{A} = \mathcal{P}_n \setminus A$.

LEMMA 2.3 *If $(\mathcal{A}, \mathcal{B})$ is $\mathcal{R}_2$, then $\mathcal{B} \subseteq \bar{\mathcal{A}}$.*

Indeed, if $B \in \mathcal{B}$, then $\bar{B} \notin \mathcal{A}$ by $\mathcal{R}_2$, and then $B \in \bar{\mathcal{A}}$ by definition of the dual adversary structure.

3 Secret sharing

Our protocols use secret sharing, a well-known cryptographic primitive introduced by Shamir [Sha79] and, independently, Blakley [Bla79]. We recall some terminology and results which will be needed later.

Let $\mathcal{S}$ be a secret sharing scheme for the set of players $\mathcal{P}_n$. We say that a set $A \subseteq \mathcal{P}_n$ is unqualified if the set of shares corresponding to A gives no information about the secret. Note that the family $\mathcal{A} \subseteq 2^{\mathcal{P}_n}$ of all unqualified sets of $\mathcal{S}$ is an adversary structure. A set $A \subseteq \mathcal{P}_n$ is qualified if the set of shares corresponding to A uniquely determines the secret. The family of all qualified sets is called the access structure of $\mathcal{S}$. We say that a secret sharing scheme is perfect if every set $A \subseteq \mathcal{P}_n$ is either qualified or unqualified (there are no sets of shares which give partial information about the secret).

$\texttt{Share}_{\mathcal{S}}$ is a probabilistic algorithm that takes as input s and outputs a valid sharing for it. We also define $\texttt{Reconstruct}_{\mathcal{S}}$, an algorithm that takes as input a set of pairs $\{(i, a_i) : i \in A\}$ where $A \subseteq \mathcal{P}_n$ and for every i, a_i is an element of the space of shares corresponding to player i and outputs s if A is a qualified set for $\mathcal{S}$ and the values $\{a_i : i \in A\}$ are part of a valid sharing of the secret s, and outputs $\perp$ otherwise (note that if A is qualified, at most one secret can be consistent with the values $\{a_i : i \in A\}$).

Let $\mathbb{F}$ be a finite field. A linear secret sharing scheme $\mathcal{S}$ (over $\mathbb{F}$), LSSS for short, is one where the space of secrets is a vector space $\mathbb{F}^{\ell_0}$, the space of the i-th shares is $\mathbb{F}^{\ell_i}$ for $i = 1, \ldots, n$, and the shares are computed as a linear function of the secret and a uniformly random vector $\mathbf{u} \in \mathbb{F}^e$. We denote by $[s, \mathbf{u}]_{\mathcal{S}} \in \mathbb{F}^{\ell}$ the vector resulting of concatenating the shares of secret s with randomness $\mathbf{u}$, where $\ell = \sum_{i=1}^{n} \ell_i$. When we do not need to make the randomness explicit, then we write $[s]_{\mathcal{S}}$. Moreover, we say that ℓ is the complexity of the LSSS. We note that $\texttt{Share}_{\mathcal{S}}$ runs in polynomial time in ℓ. It is easy to see that we can define an algorithm $\texttt{Reconstruct}_{\mathcal{S}}$, based on solving systems of linear equations, that runs in polynomial time in ℓ.

It is a well known result [ISN87] that every adversary structure is the adversary structure of a LSSS.

THEOREM 3.1 *For every finite field $\mathbb{F}$ and integer $\ell_0 \geq 1$ and for every adversary structure $\mathcal{A}$ there exists a perfect LSSS $\mathcal{S}_{\mathcal{A}}$ with secrets in $\mathbb{F}^{\ell_0}$ and adversary structure $\mathcal{A}$.*

In general the complexity of the LSSS $\mathcal{S}_{\mathcal{A}}$ in [ISN87] is exponential in n.

We say that a LSSS is ideal if $\ell_i = 1$ for all i.[2] The complexity of an ideal LSSS is n, which is smallest possible. Given a field $\mathbb{F}$ and an adversary structure $\mathcal{A}$, it is not necessarily true that there exists an ideal LSSS over $\mathbb{F}$ with $\mathcal{A}$ as its adversary structure. In fact, there are families of adversary structures $\mathcal{A}$ such that for any finite field $\mathbb{F}$, the smallest complexity of an LSSS with $\mathcal{A}$ as its adversary structure is superpolynomial in n. See [Bei11] for more information.

4 Security Model

Our model is the client-server model. We have two clients who wish to realize secure computation with the aid of n servers. Each client can corrupt a certain set of servers and its corruption capability

[2] We include the case in which some shares are dummy, i.e., always zero.

is defined by an adversary structure. In this paper, we ignore the case where servers are corrupted by an entity which is not one of the players. In our protocol, we will first consider cases where only one player is malicious and corrupts servers while the other player is honest and does not corrupt servers. We will prove security of our protocol in the *Universal Composability (UC)* framework, introduced by Canetti [Can01]. We will then also consider the case where they both are corrupted, one semi-honestly and the other either in a malicious or semi-honest fashion, and in addition both may corrupt servers. We use the *Universal Composability with Local Adversaries* framework (abbreviated by Local Universal Composability or LUC), introduced by Canetti and Vald [CV12], to prove security in those cases.

Universal Composability is based on the simulation paradigm. Roughly, the idea is to compare the execution of the actual protocol (the real world) with an idealized scenario (the ideal world) in which the computations are carried out by a trusted third party (the ideal functionality) which receives inputs from and hands in outputs to the players. The goal is to show that these two worlds are indistinguishable. In order to formalize this goal, we introduce a party called the environment $\mathcal{Z}$, whose task is to distinguish between both worlds. Furthermore, in the ideal world, we introduce a simulator Sim, its task being to simulate any action of the adversary in the real protocol and thereby to make the two views indistinguishable for any environment. More precisely, in the real world execution of protocol π, with the adversary Adv and environment $\mathcal{Z}$, the environment provides input and receives output from both Adv and π. Call $\mathtt{Real}_{\mathrm{Adv},\pi,\mathcal{Z}}$ the view of $\mathcal{Z}$ in this execution. In the ideal world $\mathcal{Z}$ provides input and receives output from Sim and the ideal functionality $\mathcal{F}$. Call $\mathtt{Ideal}_{\mathrm{Sim},\mathcal{F},\mathcal{Z}}$ the view of $\mathcal{Z}$ in the ideal execution. We can proceed to define what it means for a protocol to be secure.

DEFINITION 4.1 *A protocol π UC-implements a functionality $\mathcal{F}$ against a certain class of adversaries $\mathcal{C}$ if for every adversary* $\mathrm{Adv} \in \mathcal{C}$ *there exists a simulator* Sim *such that for every environment $\mathcal{Z}$, $\mathtt{Real}_{\mathrm{Adv},\pi,\mathcal{Z}} \approx \mathtt{Ideal}_{\mathrm{Sim},\mathcal{F},\mathcal{Z}}$.*

The cornerstone of the universal composability framework is the composition theorem, which works as follows. Denote by $\pi \circ G$ a protocol π that during its execution makes calls to an ideal functionality G. The composition proof shows that if $\pi_f \circ G$ securely implements $\mathcal{F}$ and if π_g securely implements G then $\pi_f \circ \pi_g$ securely implements $\mathcal{F}$. This provides modularity in construction of protocols and simplifies proofs dramatically. It is also shown that proving security against a dummy adversary, one who acts as a communication channel, is sufficient for proving general security.

Universal Composability as we have described it so far considers a single real-world adversary which corrupts parties and chooses their behaviour to attack the protocol and a single ideal-world simulator which generates a view consistent for the given real world adversary. However, this notion does not capture the case where there are more than one "local" adversaries which do not work together or more precisely do not share a view of the system. This means that Universal Composability does not allow us to deal with certain notions of security such as collusion freeness, anonymity, deniability or security in game-theoretic scenarios.

To capture such a notion, Canetti and Vald [CV12] defined the notion of Local Universal Composability. Roughly speaking, instead of having a single adversary which corrupts participants, there are multiple adversaries, each of whom can only corrupt a single participant. In the ideal world, each adversary will be replaced by a simulator. The simulators can only communicate with each other either through the environment or through the ideal functionality. Local universal composability also considers hybrid models.

Canetti and Vald describe the power of their notion as follows: *"If π is a LUC-secure protocol that implements a trusted party $\mathcal{F}$ then each individual entity participating in π affects each other entity in the system no more than it does so in the ideal execution with $\mathcal{F}$."* A general composition theorem is provided which allows the replacement of an ideal functionality with a protocol which implements it. Moreover, it is also proven that security with respect to dummy adversaries implies security against a general adversary. We denote the parties as $\{P_i : i \in I\}$. As mentioned above, to each party P_i corresponds an adversary Adv_i and a simulator Sim_i. Let $\mathcal{C}$ be a class of tuples of adversaries $(\mathrm{Adv}_i)_{i\in I}$.

DEFINITION 4.2 *π LUC-implements $\mathcal{F}$ against the class $\mathcal{C}$ if for every $(\mathrm{Adv}_i)_{i \in I} \in \mathcal{C}$ and every $\bar{I} \subseteq I$, there exists a simulator $\mathrm{Sim} = \cup_{i \in \bar{I}} \mathrm{Sim}_i$ such that for every environment $\mathcal{Z}$, we have that $\mathtt{Ideal}_{\mathrm{Sim},\mathcal{F},\mathcal{Z}} \approx \mathtt{Real}_{\mathrm{Adv},\pi,\mathcal{Z}}$, where $\mathrm{Adv} = \cup_{i \in \bar{I}} \mathrm{Adv}_i$.*

In our case, since we consider the possibility where both players are semi-honest, it must be the case that the simulators are able to get shares to each other. Since the simulators can only communicate to each other via the ideal functionality or the environment and the environment is untrustworthy, it must be the case that these values can be extracted from the ideal functionality.

5 A protocol for semi-honest adversaries

As a warm-up, we sketch a simple protocol that allows secure computation for semi-honest adversaries, simultaneously corrupting adversary structures $\mathcal{A}$ and $\mathcal{B}$, as long as $(\mathcal{A}, \mathcal{B})$ is $\mathcal{R}_2$. First, we do not claim any novelty with regard to feasibility in this case, since one could obtain a protocol in a straightforward way from the general results on MPC secure against semi-honest adversaries in [HM00] (which is secure as long as the adversary corrupts a $\mathcal{Q}_2$ structure) applied to the $n + 2$ players consisting of Alice, Bob and the servers. While in that protocol servers would communicate to each other, we can emulate such communications in our setting by having the sending server split additively its message in two halves and communicating one share to the receiving server via Alice and the other via Bob (so that neither Alice nor Bob learn anything new about the message).

However, this requires the communication complexity to be at least square in n, the number of servers (the actual complexity depending on the complexity of the secret sharing scheme). In contrast, we show a protocol whose communication complexity (in n) is essentially given by the complexity of the secret sharing scheme for $\mathcal{A}$ and, hence, it can be as low as linear in n.

We introduce the following notation: for two vectors $\mathbf{a}, \mathbf{b}$ of the same length, their coordinatewise product is denoted $\mathbf{a} * \mathbf{b}$ and their outer (or Kronecker) product is denoted $\mathbf{a} \otimes \mathbf{b}$. Let $\mathcal{S}_{\mathcal{A}}$ be a perfect LSSS for adversary structure $\mathcal{A}$ and with secrets in a finite field $\mathbb{F}$, according to Theorem 3.1. It follows from a construction in [CDM00] that from $\mathcal{S}_{\mathcal{A}}$ we can build an LSSS $\bar{\mathcal{S}}_{\mathcal{A}}$ for $\bar{\mathcal{A}}$, and that furthermore, for any secrets s, s', the product ss' can be reconstructed as the sum of the coordinates of the vector $[s, \mathbf{u}]_{\mathcal{S}_{\mathcal{A}}} * [s', \mathbf{v}]_{\bar{\mathcal{S}}_{\mathcal{A}}}$. Note that in [CDM00], it is assumed that the adversary structure is $\mathcal{Q}_2$, but this assumption is only used there to guarantee that $\bar{\mathcal{A}}$ is contained in $\mathcal{A}$, which we do not need and the above product reconstruction property holds even if $\mathcal{A}$ is not $\mathcal{Q}_2$. The idea of the protocol is now as follows: Bob believes that Alice may corrupt a set of servers that is in $\mathcal{A}$ (but nothing more), so he is happy to share a secret among the servers using $\mathcal{S}_{\mathcal{A}}$. Alice, on the other hand, believes that Bob may corrupt a set in $\mathcal{B}$, but by the $\mathcal{R}_2$ condition and Lemma 2.3, we have that $\mathcal{B} \subseteq \bar{\mathcal{A}}$, so Alice will be happy to share a secret among the servers using $\bar{\mathcal{S}}_{\mathcal{A}}$. This and the product reconstruction property will imply that we can implement multiplication of a value from Alice and one from Bob.

We will represent a secret value $x \in \mathbb{F}$ in the computation in additively shared form: we will write $< x >$ to denote a pair (x_A, x_B) with $x = x_A + x_B$ where Alice holds x_A and Bob holds x_B. Usually the pair will be randomly chosen such that the sum is x, but we suppress the randomness from the notation for simplicity. We define the sum $< x > + < y >$ via component wise addition to be the pair $(x_A + y_A, x_B + y_B)$ and multiplication by a public constant similarly. Then we trivially have $< x > +\alpha < y >=< x + \alpha y >$.

We then define a protocol (see Figure 1) for computing securely an arithmetic circuit over $\mathbb{F}$; the parties define representations of their inputs and then work their way through the circuit using the subprotocols for addition and multiplication, as appropriate. In the end the outputs are revealed.

It is trivial to see that this protocol computes correct results since both parties follow the protocol. Informally, privacy holds because one party always uses a secret sharing scheme for which the other party can only corrupt an unqualified set. Furthermore, the set of n values received by Bob in the Multiplication subprotocol is easily seen to be a set of uniformly random values that reveal no side information.

Semi-honest Secure Protocol.

Input If Alice holds input x, we define $< x >= (x, 0)$, if Bob holds y, we define $< y >= (0, y)$.

Addition Given $< a >, < b >$, Alice and Bob compute $< a > + < b >=< a + b >$ by local computation.

Multiplication by constant Given α and $< a >$, Alice and Bob compute $\alpha < a >=< \alpha a >$ by local computation.

Multiplication Subprotocol Assuming Alice holds a and Bob holds b, we want to compute a random representation $< ab >$ without revealing any information on a or b. Alice creates a set of shares $[a, \mathbf{u}]_{\bar{\mathcal{S}}_A}$ for random $\mathbf{u}$ and sends the i-th share to S_i. Similarly Bob creates and distributes $[b, \mathbf{v}]_{\mathcal{S}_A}$. Finally Alice chooses a random $r \in \mathbb{F}$ and random $r_1, ..., r_n \in \mathbb{F}$ subject to $r = r_1 + ... + r_n$, and sends r_i to S_i.

Let a_i, b_i be the shares of a, b received by S_i. He now computes $w_i = a_i \cdot b_i - r_i$ (where $a_i \cdot b_i \in \mathbb{F}$ denotes the inner product of a_i and b_i) and sends it to Bob, who computes $\sum_i w_i$. The final representation is defined to be $< ab >= (r, \sum_i w_i)$.

Multiplication Given $< x >= (x_A, x_B), < y >= (y_A, y_B)$, we want to compute a representation $< xy >$. Execute the above Multiplication subprotocol twice. First, with $(a, b) = (x_A, y_B)$ to get $< x_A y_B >= (a_1, b_1)$ and second, with $(a, b) = (y_A, x_B)$ to get $< y_A x_B >= (a_2, b_2)$. Then we define the output to be

$$< xy >= (x_A y_A + a_1 + a_2, x_B y_B + b_1 + b_2)$$

Figure 1: Semi-honest Secure Protocol

One might imagine that the semi-honest secure protocol could be made secure against malicious adversaries in the standard way, i.e., by using verifiable secret sharing to commit parties to the values they hold and in this way allow them to prove that the protocol was followed. However, verifiable secret sharing requires (at least) that the adversary structure used is $\mathcal{Q}_2$ and as we have seen this may not be satisfied for any of the two adversary structures.[3] In addition, the use of verifiable secret sharing would also introduce the necessity of communication between servers, so we would not attain our goal of minimal connectivity either. Therefore, we follow a different approach in the following section.

6 The protocol

In this section we present our main oblivious transfer protocol. We assume again that the adversary structures $\mathcal{A}$ and $\mathcal{B}$ satisfy that $(\mathcal{A}, \mathcal{B})$ is $\mathcal{R}_2$.

To simplify the exposition, we first assume that $\mathcal{A}$ is the adversary structure of a perfect *ideal* linear secret sharing scheme $\mathcal{S}$ over the field of two elements $\mathbb{F}_2 = \{0, 1\}$, and in the Appendix we give a more general version of our protocol that does not impose this "ideality" requirement on $\mathcal{A}$.

Let $\mathcal{M}$ be the space of messages. Without loss of generality assume $\mathcal{M} = \mathbb{F}_2^m$ for some positive integer m. We fix a default message $\mathbf{0} \in \mathcal{M}$ (for example if we see the elements of $\mathcal{M}$ as bit strings of certain length, we can take $\mathbf{0}$ to be the all-zero string).

Given $\mathcal{S}$, we construct two other perfect LSSS that we call $\mathcal{S}_0$ and $\mathcal{S}_1$. The space of secrets is in both cases $\mathcal{M}$ and they are both defined on a set of $2n$ players indexed by the set $\mathcal{P}_{n,2} := \{(i, j) : i \in \{1, \ldots, n\}, j \in \{0, 1\}\}$. Each scheme $\mathcal{S}_i$ will realize an access structure Γ_i defined from the scheme $\mathcal{S}$ as follows.

Let $V_0 = \{[0, \mathbf{u}]_{\mathcal{S}} : \mathbf{u} \in \mathbb{F}_2^e\} \subseteq \mathbb{F}_2^n$ be the set of all possible sharings of 0 with the scheme $\mathcal{S}$. That is, $\mathbf{v} = (v_1, \ldots, v_n)$ belongs to V_0 if and only if there exists a sharing of 0 with $\mathcal{S}$ where each player i receives v_i. Now, the minimally qualified sets of Γ_0 (the access structure to be realized by $\mathcal{S}_0$) are exactly the sets $\{(1, v_1), \ldots, (n, v_n)\} \subseteq \mathcal{P}_{n,2}$ with $(v_1, \ldots, v_n) \in V_0$. Obviously this means that every set $\{(1, w_1), \ldots, (n, w_n)\}$ with $(w_1, \ldots, w_n) \notin V_0$ is unqualified. Similarly, let $V_1 = \{[1, \mathbf{u}]_{\mathcal{S}} : \mathbf{u} \in$

[3] Note that in the protocol above, secret sharing is always performed with one of the clients as the dealer and the servers as the players who receive shares, the other client not taking part.

$\mathbb{F}_2^e\} \subseteq \mathbb{F}_2^n$ be the set of all possible sharings of 1 with $\mathcal{S}$. The minimally qualified sets in Γ_1 are the sets $\{(1, v_1), \ldots, (n, v_n)\} \subseteq \mathcal{P}_{n,2}$ with $(v_1, \ldots, v_n) \in V_1$.

The existence of $\mathcal{S}_0$ and $\mathcal{S}_1$ is guaranteed by Theorem 3.1. However, we can in fact show a much stronger (in terms of efficiency) result, which we prove in Appendix B:

PROPOSITION 6.1 *For any ideal linear secret sharing scheme $\mathcal{S}$ over $\mathbb{F}_2$, there exist two* ideal *secret sharing schemes $\mathcal{S}_0$ and $\mathcal{S}_1$ respectively realizing Γ_0 and Γ_1 defined from $\mathcal{S}$ as above.*

We now construct the protocol π_{OT} as described in Figure 2.

Oblivious transfer protocol π_{OT}

1. Alice independently creates sharings $[m_0]_{\mathcal{S}_0} = (m_0^{(i,j)})_{i\in\{1,\ldots,n\},j\in\{0,1\}}$ and $[m_1]_{\mathcal{S}_1} = (m_1^{(i,j)})_{i\in\{1,\ldots,n\},j\in\{0,1\}}$ for her inputs.
   Bob creates a sharing $[b]_{\mathcal{S}} = (b_1, \ldots, b_n)$ of his input. Note that each $b_i \in \{0, 1\}$ because $\mathcal{S}$ is ideal.
   During the protocol, the servers S_i ignore any message that is not of the form specified by the protocol.
2. Bob sends $(\texttt{Bob-share}, i, b_i)$ to server S_i.
3. S_i sends $(\texttt{ready}, i)$ to Alice.
4. After Alice has received $(\texttt{ready}, i)$ from every server S_i, she sends the messages $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ to each server S_i where $u_i^0 := m_0^{(i,0)}||m_1^{(i,0)}$ and $u_i^1 = m_0^{(i,1)}||m_1^{(i,1)}$.
5. Server S_i sends $(\texttt{output}, i, u_i^{b_i})$ to Bob.
6. If for any i, $u_i^{b_i} \notin \mathcal{M}^2$, then Bob outputs $\mathbf{0}$. Otherwise, Bob parses each $u_i^{b_i}$ as $m_0^{(i,b_i)}||m_1^{(i,b_i)}$.
   If $b = 0$, Bob constructs m_0 by applying $\texttt{Reconstruct}_{\mathcal{S}_0}(\{((i, b_i), m_0^{(i,b_i)}) : i \in \mathcal{P}_n\})$.
   If $b = 1$, Bob constructs m_1 by applying $\texttt{Reconstruct}_{\mathcal{S}_1}(\{((i, b_i), m_1^{(i,b_i)}) : i \in \mathcal{P}_n\})$.
   In any of the cases, if the reconstruction fails, output $\mathbf{0}$. Otherwise output the reconstructed m_b.

Figure 2: Protocol π_{OT}

Note that in the protocol π_{OT}, if we ignore the headers of the messages (i.e., the command names and number of the server involved) the communication complexity of π_{OT} is as follows: Alice sends $4m$ bits to each server, in total $4nm$ bits. Bob sends one bit to each server, in total n bits. Finally each server sends $2m$ bits to Bob. The total amounts of bits communicated is hence $(6m + 1)n$ bits.

PROPOSITION 6.2 *If Alice and Bob follow the protocol honestly, then π_{OT} implements OT correctly.*

PROOF. If Alice and Bob follow the protocol honestly, at the end of the protocol Bob will have received all values $m_b^{(i,b_i)}, i = 1, \ldots, n$, for some sharing $[b]_{\mathcal{S}} = (b_1, \ldots, b_n)$. By definition the set $\{(1, b_1), \ldots, (n, b_n)\}$ is qualified for $\mathcal{S}_b$ (because $(b_1, \ldots, b_n) \in V_b$) and hence Bob has enough information to reconstruct m_b. Hence, the protocol is correct if both Alice and Bob are (semi-) honest. △

7 Security

We now prove security of this protocol, in two steps: first, we show security against one adversary in the UC model, and then we will consider security against simultaneous adversaries in the LUC model.

7.1 Security in the UC model

We will first show that this protocol implements securely the functionality $\mathcal{F}_{OT}$ described in Figure 3 in the Universal Composability framework. This will serve both as a warm up and a reference when we prove security in the Local Universal Composability framework later on.

Functionality $\mathcal{F}_{OT}$

1. On input $(\texttt{transfer}, b)$ from Bob, send $(\texttt{ready})$ to Alice.
2. On input $(\texttt{send}, m_0, m_1)$ from Alice, if $(\texttt{transfer}, b)$ has been received previously from Bob, send $(\texttt{sent}, m_b)$ to Bob.

Figure 3: Functionality $\mathcal{F}_{OT}$

THEOREM 7.1 *Let $(\mathcal{A}, \mathcal{B})$ be an $\mathcal{R}_2$ pair of structures, and assume that $\mathcal{A}$ admits an ideal linear secret sharing scheme. Consider the class of all adversaries that either corrupt Alice together with a set $A \in \mathcal{A}$ or Bob together with a set $B \in \mathcal{B}$. Then the protocol π_{OT} UC-implements the functionality $\mathcal{F}_{OT}$ against that class of adversaries.*

PROOF.

Alice honest, Bob malicious: The idea is to have the simulator extract the environment's input by applying the reconstruction procedure of $\mathcal{S}$ to the shares b_i received from the environment. By the $\mathcal{R}_2$ property, these shares can be consistent with at most one b. If in fact Bob is bound to a bit b, this is sent to the functionality and m_b is received; the simulator then sets m_{1-b} at random and generates shares for both messages. Otherwise, the simulator generates shares for random m_0, m_1.

We need two lemmas. The first uses the $\mathcal{R}_2$ property to argue that the set of servers not corrupted by Bob is qualified in the scheme $\mathcal{S}$ and therefore no matter what he sends to the uncorrupted servers, this can be consistent with at most one possible input.

LEMMA 7.2 *If $(\mathcal{A}, \mathcal{B})$ is an $\mathcal{R}_2$ pair of structures, and $\mathcal{S}$ is a perfect secret sharing scheme with $\mathcal{A}$ as its adversary structure, then for every $B \in \mathcal{B}$, its complement $\overline{B}$ is qualified in $\mathcal{S}$.*

This is because by definition of $\mathcal{R}_2$, $\overline{B} \notin \mathcal{A}$. Our next lemma will guarantee the privacy of Alice's input.

LEMMA 7.3 *Let m_0, m_1 be shared independently with $\mathcal{S}_0$, $\mathcal{S}_1$ respectively. Fix $B \subseteq \{1, \ldots, n\}$ and $(b'_1, \ldots, b'_n) \in \mathbb{F}_2^n$, and define $\mathcal{I}' = \{(i, b'_i) : i \in \overline{B}\} \cup \{(i, j) : i \in B, j \in \{0, 1\}\}$.*
Fix $b \in \{0, 1\}$. If the set $\{b'_i : i \in \overline{B}\}$ is not part of any sharing $[b]_\mathcal{S}$ then the values $m_0^{(i,j)}$, $m_1^{(i,j)}$, $(i, j) \in \mathcal{I}'$ give no information about m_b.

PROOF. Since the sharings of m_0 and m_1 are independent, clearly the shares of m_{1-b} cannot add information about m_b. Hence, we need to prove that the shares $m_b^{(i,j)}$, $(i, j) \in \mathcal{I}'$ give no information about m_b, i.e., that the set $\mathcal{I}'$ is unqualified for $\mathcal{S}_b$. But if $\mathcal{I}'$ were qualified for $\mathcal{S}_b$, it would contain a set $\{(1, b_1), \ldots, (n, b_n)\} \subseteq \mathcal{P}_{n,2}$ with $(b_1, \ldots, b_n) \in V_b$. However then necessarily $b_i = b'_i$ for all $i \in \overline{B}$ and that would mean $\{b'_i : i \in \overline{B}\}$ belongs to a sharing $[b]_\mathcal{S}$ which contradicts the assumption. △

We now describe the simulator Sim. We will suppose without loss of generality that corrupted servers act as a dummy adversary. Let B denote the set of corrupted servers.

First, Sim awaits $(\texttt{ready}, i)$ for $i \in B$ and that the environment has sent b_i for each $i \in \overline{B}$. Then Sim executes $\texttt{Reconstruct}_\mathcal{S}(\{(i, b_i) : i \in \overline{B}\})$. If the reconstruction fails then Sim chooses random messages $\tilde{m}_0$, $\tilde{m}_1$. If the reconstruction succeeds, let b be its output; then Sim sends the command $(\texttt{transfer}, b)$ to $\mathcal{F}_{OT}$, receives message $(\texttt{sent}, m_b)$ and sets $\tilde{m}_b := m_b$; it selects a random message $\tilde{m}_{1-b} \in \mathcal{M}$.

In any case, Sim generates shares for $\tilde{m}_0$ using $\mathcal{S}_0$ and shares for $\tilde{m}_1$ using $\mathcal{S}_1$. It creates the values $u_i^0 := \tilde{m}_0^{(i,0)} || \tilde{m}_1^{(i,0)}$ and $u_i^1 = \tilde{m}_0^{(i,1)} || \tilde{m}_1^{(i,1)}$. Finally, in parallel Sim sends the following to the environment: for each $i \in \overline{B}$, he sends $(\texttt{output}, i, u_i^{b_i})$ and for each $i \in B$, he sends $(\texttt{Alice-share}, i, u_0^i, u_1^i)$.

In order to prove indistinguishability, we should first note that, by Lemma 7.2, the set $\overline{B}$ is qualified for $\mathcal{S}$ and hence, the values $\{b_i : i \in \overline{B}\}$ cannot be part of both a sharing $[0]_\mathcal{S}$ and a sharing $[1]_\mathcal{S}$. It is now easy to see, by Lemma 7.3, that the distribution of shares received by $\mathcal{Z}$ in the simulation is indistinguishable from the distribution of shares received in the real world.

Alice malicious, Bob honest: The simulation in this case is slightly tricky, since a potential problem of the protocol is that Alice can generate inconsistent shares which make Bob's output dependent on his selections (that is, on the random sharing of his input). We show, perhaps surprisingly, that this does not affect the security of the protocol. Essentially, the simulator will generate one sharing for $b = 0$ and one for $b = 1$ such that the shares corresponding to the corrupted servers coincide. The simulator will then construct the value that a receiver would construct for each of these two sharings and will send these values to the functionality. This results in a view in the ideal world which is perfectly indistinguishable from the real world, due to the privacy for the set of corrupted servers.

We will suppose without loss of generality that corrupted servers act as a dummy adversary. Let $A \in \mathcal{A}$ be the set of corrupted servers. The simulator works as follows:

Upon receiving $(\texttt{ready})$ from the ideal functionality $\mathcal{F}_{OT}$, Sim generates uniformly random sharings of $b = 0$ and $b' = 1$ in $\mathcal{S}$ subject to the only condition that if $i \in A$, then $b_i = b'_i$. Note that this is possible since A is unqualified for $\mathcal{S}$. Then, in parallel Sim sends to the environment the message $(\texttt{ready}, i)$ for each i and the message $(\texttt{Bob-share}, i, b_i)$ for each $i \in A$. Sim now awaits that for each $i \in \overline{A}$, the environment sends u_i^0 and u_i^1 and that for each $i \in A$ the environment sends $u_i^{b_i}$. If any u_i^j is not an element of $\mathcal{M}^2$, then, Sim does the following: if $b_i = j$, set $m_0 = \mathbf{0}$, and if $b'_i = j$, set $m_1 = \mathbf{0}$. For the rest of the u_i^j, Sim does the following: Sim parses, for $i \in \overline{A}$, u_i^0 as $m_0^{(i,0)}||m_1^{(i,0)}$ and u_i^1 as $m_0^{(i,1)}||m_1^{(i,1)}$. Sim also parses, for $i \in A$, $u_i^{b_i}$ as $m_0^{(i,b_i)}||m_1^{(i,b'_i)}$ (again, note $b_i = b'_i$ for $i \in A$).

For $k = 0, 1$, if m_k is not already set to $\mathbf{0}$ then Sim computes

$$m_k = \texttt{Reconstruct}_{\mathcal{S}_k}(\{((i, b_i), m_k^{(i,b_i)}) : i \in \mathcal{P}_n\})$$

If the reconstruction of m_k fails, Sim sets $m_k = \mathbf{0}$. Finally, it sends $(\texttt{send}, m_0, m_1)$ to $\mathcal{F}_{OT}$.

By construction, the shares b_i corresponding to the set A of corrupt servers that the environment receives are indistinguishable from the A-shares in a uniformly random sharing of b, regardless of whether $b = 0$ or $b = 1$. Hence these b_i do not allow the receiver to distinguish the real and ideal world. Now, since after that step there is no further interaction, it suffices to show that the messages sent to Bob are indistinguishable from the ones sent in the real world.

This is the case since the shares have been chosen with the distribution Bob would use and since the simulator reconstructs the messages m_0 and m_1 in exactly the same way as Bob would reconstruct m_b in the real protocol, if b is his input. Therefore the real and ideal world are indistinguishable. $\triangle$

We note that the simulators in the proof above run in polynomial time, because $\mathcal{S}_0$ and $\mathcal{S}_1$ are ideal.

Finally, we remark that the Oblivious Transfer protocol we have presented can easily be extended to the case where there does not exist an ideal secret sharing scheme for Alice's adversary structure. We give a complete description of the protocol in Appendix C.

7.2 Local Universal Composability

In this section, we discuss the security of our protocol in the Local Universal Composability model. We first define the functionality that we want to implement securely. We denote the possible degrees of corruption by $C = \{\text{Malicious, Semi-honest, Honest}\}$. The functionality $\mathcal{F}_{OT}^{L}$ will be the composition of three ideal functionalities: one center box, denoted by $\mathcal{F}_{CIOT}$ (Figure 6), and two outer-boxes, denoted by $\mathcal{I}_{\text{HHA}}$ (Figure 4) and $\mathcal{I}_{\text{HHB}}$ (Figure 5) respectively. The local simulator Sim_A for $\mathbb{A}$ (respectively Sim_B for $\mathbb{B}$) will communicate with $\mathcal{I}_{\text{HHA}}$ (respectively $\mathcal{I}_{\text{HHB}}$) only.

Each of the outer boxes will know the level of corruption of both players. $\mathcal{I}_{\text{HHA}}$ will learn the level of corruption of Alice directly from the local simulator Sim_A, while it will learn the level of corruption of Bob via the functionality $\mathcal{F}_{CIOT}$. The same (but with the roles swapped) will hold for $\mathcal{I}_{\text{HHB}}$.

The goal of the outer boxes is to hide from the local simulators whether the other party is honest, semi-honest or malicious (we use the acronym HH to denote honesty-hiding). This is done because having a functionality which would reveal the corruption level of the simulator would be useless for constructing protocols. This means that the outer boxes must simulate the case of a semi-honest party when the party in question is honest. A case-by-case (according to the corruption levels c_A and c_B) description of $\mathcal{F}_{OT}^{L}$ can be found in Appendix D.

Functionality $\mathcal{I}_{\mathrm{HHA}}$

- It awaits $(\texttt{corrupt}, c_A, A)$ from Sim_A, where $c_A \in C$, and forwards it to $\mathcal{F}_{CIOT}$. It then awaits $(\texttt{corrupt}, c_B)$ from $\mathcal{F}_{CIOT}$.
- If $c_B \neq$ Honest or $c_A = c_B =$ Honest, act as a communication channel between Sim_A and $\mathcal{F}_{CIOT}$.
- Otherwise (if $c_A \neq$ Honest and $c_B =$ Honest), on input $(\texttt{ready})$ from $\mathcal{F}_{CIOT}$:
  - It selects $b' \in \{0,1\}$ uniformly at random and generates a sharing $[b']_{\mathcal{S}} = (b'_i)_{i \in \mathcal{P}_n}$.
  - For each $i \in A$ it sends $(\texttt{Bob-share}, i, b'_i)$ to Sim_A. For each $i \notin A$, it sends $(\texttt{ready}, i)$ to Sim_A.
  - On receipt of $(\texttt{send}, m_0, m_1)$ from Sim_A, it forwards it to $\mathcal{F}_{CIOT}$.

Figure 4: Functionality $\mathcal{I}_{\mathrm{HHA}}$

Functionality $\mathcal{I}_{\mathrm{HHB}}$

- On input $(\texttt{corrupt}, c_B, B)$ from Sim_B where $c_B \in C$, and forwards it to $\mathcal{F}_{CIOT}$. It then awaits $(\texttt{corrupt}, c_A)$.
- If $c_A \neq$ Honest or $c_A = c_B =$ Honest, act as a communication channel between $\mathcal{F}_{CIOT}$ and Sim_B.
- Otherwise (if $c_A =$ Honest and $c_B \neq$ Honest):
  - It awaits $(\texttt{Bob-share}, i, b_i)$ for all i from Sim_B.
  - On receipt of $(\texttt{transfer}, b)$ from Sim_B, it forwards it to $\mathcal{F}_{CIOT}$.
  - On receipt of $(\texttt{sent}, m_b)$ from $\mathcal{F}_{CIOT}$, it selects a random $m'_{1-b} \in \mathcal{M}$ and generates random sharings $[m_b]_{\mathcal{S}_b} = (m_b^{(i,j)})_{(i,j) \in \mathcal{P}_{n,2}}$ and $[m'_{1-b}]_{\mathcal{S}_{1-b}} = ({m'}_{1-b}^{(i,j)})_{(i,j) \in \mathcal{P}_{n,2}}$.
    It creates the concatenations u_i^j, $(i,j) \in \mathcal{P}_{n,2}$ as it would happen in the protocol, i.e., $u_i^j = m_b^{(i,j)} || {m'}_{1-b}^{(i,j)}$ if $b = 0$ and $u_i^j = {m'}_{1-b}^{(i,j)} || m_b^{(i,j)}$ if $b = 1$).
    For each $i \in B$, it sends $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ to Sim_B and for each $i \notin B$, it sends $(\texttt{output}, i, u_i^{b_i})$ to Sim_B.

Figure 5: Functionality $\mathcal{I}_{\mathrm{HHB}}$

We will now argue that this ideal functionality is indeed one that we want to implement securely. Consider first the usual scenario where one of the players is honest. If both are honest then the functionality just implements OT as usual. If one of the players is honest and the other is not, the non-honest side obtains certains shares belonging to a sharing of some random element (and therefore not related to the inputs) plus, in the case where Alice is honest and Bob is semi-honest, part of a sharing of the output, but the output is already known by Bob.

Now we consider the case where none of the players are honest. In this case, our model forces us to communicate to the local simulators part of a "real" sharing of the actual inputs of the other player. This is because the information that the environment will receive from both Sim_A and Sim_B has to be consistent (as it happens to be in the real world). Note that, however, the sets of received shares give information about at most one of the inputs of Alice, and are information theoretically independent of the rest of inputs. Also note that, in the malicious case, more shares are leaked to the non-malicious side, but this is ok because we cannot guarantee privacy for the malicious player.

Functionality $\mathcal{F}_{CIOT}$

- On input $(\texttt{corrupt}, c_A, A)$ from $\mathcal{I}_{\text{HHA}}$, where $c_A \in C$:
  The ideal functionality checks that $A \in \mathcal{A}$. If $c_A =$ Honest, it also checks that $A = \emptyset$. If some of these checks fails, then it ignores further commands.

  Otherwise, it stores (c_A, A).
- On input $(\texttt{corrupt}, c_B, B)$ from $\mathcal{I}_{\text{HHB}}$, where $c_B \in C$:
  The ideal functionality checks that $B \in \mathcal{B}$. If $c_B =$ Honest, it also checks that $B = \emptyset$. If some of these checks fails, then it ignores further commands.

  Otherwise, it stores (c_B, B).
- The ideal functionality sends $(\texttt{corrupt}, c_A)$ to $\mathcal{I}_{\text{HHB}}$ and $(\texttt{corrupt}, c_B)$ to $\mathcal{I}_{\text{HHA}}$.
- On input $(\texttt{transfer}, b)$ from $\mathcal{I}_{\text{HHB}}$, send $(\texttt{ready})$ to $\mathcal{I}_{\text{HHA}}$.
  On input $(\texttt{send}, m_0, m_1)$ from $\mathcal{I}_{\text{HHA}}$, if $(\texttt{transfer}, b)$ has been received previously from $\mathcal{I}_{\text{HHB}}$, send $(\texttt{sent}, m_b)$ to $\mathcal{I}_{\text{HHB}}$.
- If $c_A =$ Semi-honest:
  On command $(\texttt{Bob-share}, i, b_i)$ from $\mathcal{I}_{\text{HHB}}$: if $i \in A$ it sends $(\texttt{Bob-share}, i, b_i)$ to $\mathcal{I}_{\text{HHA}}$; otherwise, it sends $(\texttt{ready}, i)$ to $\mathcal{I}_{\text{HHA}}$.
- If $c_B =$ Semi-honest:
  On command $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ from $\mathcal{I}_{\text{HHA}}$: if $i \in B$, it sends $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ to $\mathcal{I}_{\text{HHB}}$; otherwise, it sends $(\texttt{output}, i, u_i^{b_i})$ to $\mathcal{I}_{\text{HHB}}$.
- If $c_A =$ Malicious:
  Any command from $\mathcal{I}_{\text{HHA}}$ is directly forwarded to $\mathcal{I}_{\text{HHB}}$.
- If $c_B =$ Malicious:
  Any command from $\mathcal{I}_{\text{HHB}}$ is directly forwarded to $\mathcal{I}_{\text{HHA}}$.

Figure 6: Functionality $\mathcal{F}_{CIOT}$

THEOREM 7.4 *Let $(\mathcal{A}, \mathcal{B})$ be a $\mathcal{R}_2$ pair of structures and assume $\mathcal{A}$ admits an ideal LSSS over $\mathbb{F}_2$. Then π_{OT} LUC-implements $\mathcal{F}_{OT}^L$ against the class of pairs $(\mathbb{A}, \mathbb{B})$ of adversaries where $\mathbb{A}$ corrupts Alice and $A \in \mathcal{A}$ and $\mathbb{B}$ corrupts Bob and $B \in \mathcal{B}$.*

We prove this Theorem in Appendix E. Here we give an intuition. The case where one of the players, say Alice, is honest reduces to the universal composability proof with a few differences. The inner functionality $\mathcal{F}_{CIOT}$ acts as the functionality $\mathcal{F}_{OT}$ in the UC case, except it sends some additional messages which are ignored by $\mathcal{I}_{\text{HHA}}$ if Alice is honest. Furthermore we will define Bob's simulator so that its composition with $\mathcal{I}_{\text{HHB}}$ acts as the simulator for the UC case. If both Alice and Bob are corrupted, then we have the added complication that in the real world the environment will receive certain information from $\mathbb{A}$ that is consistent with the information received from $\mathbb{B}$. The simulation needs to guarantee this also happens in the ideal world, and for this we use the fact that the ideal functionality has the ability to transfer the appropriate parts of the information from one simulator to the other.

8 Acknowledgement

We are grateful to an anonymous reviewer of a previous version of this paper for insisting that we clarify the relation between our results and the single-adversary impossibility result in [FHM99], which helped us improved the presentation.

References

[Bei11] Amos Beimel. Secret-Sharing Schemes: A Survey. In *IWCC*, pages 11–46, 2011.

[Bla79] George Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, volume 48, pages 313–317, June 1979.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2000.

[CV12] Ran Canetti and Margarita Vald. Universally Composable Security with Local Adversaries. In *SCN*, pages 281–301, 2012.

[DI05] Ivan Damgård and Yuval Ishai. Constant-Round Multiparty Computation Using a Black-Box Pseudorandom Generator. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394. Springer, 2005.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A Randomized Protocol for Signing Contracts. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 205–210, 1982.

[FHM99] Matthias Fitzi, Martin Hirt, and Ueli Maurer. General adversaries in unconditional multiparty computation. In *Advances in Cryptology-ASIACRYPT99*, pages 232–246. Springer, 1999.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In *Theory of Cryptography*, pages 393–411. Springer, 2008.

[HKN$^+$05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Advances in Cryptology–EUROCRYPT 2005*, pages 96–113. Springer, 2005.

[HM00] Martin Hirt and Ueli M. Maurer. Player Simulation and General Adversary Structures in Perfect Multiparty Computation. *J. Cryptology*, 13(1):31–60, 2000.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and Jürg Wullschleger. Constant-Rate Oblivious Transfer from Noisy Channels. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 667–684. Springer, 2011.

[ISN87] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structures. In *Proc. IEEE GlobeCom '87 Tokyo*, pages 99–102, 1987.

[Kil88] Joe Kilian. Founding Cryptography on Oblivious Transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[KMR11] Semy Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing Multi-Party Computation, https://eprint.iacr.org/2011/272.pdf. 2011.

[Rab81] Michael Rabin. How to Exchange Secrets with Oblivious Transfer. Technical report, Aiken Computation Lab, Harvard University, 1981.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

A Security Against Corruptions of Only Servers

As explained in the introduction, our model does not consider corruption of only servers, and our security proofs therefore do not directly guarantee any security in case the adversaries corrupt only a set of servers. Nevertheless, we can argue that some security properties are satisfied even in case of server-only corruption.

Let us first consider the case where Alice is honest and has input x_A, while $\mathbb{B}$ is semi-honest and corrupts only a set $S \in \mathcal{B}$ of servers. So Bob is also honest and has input x_B. If a protocol π is secure in our model, it is easy to see that it will compute the correct result $f(x_A, x_B)$ also in this case and that $\mathbb{B}$ will learn nothing more than $f(x_A, x_B), x_B$. This follows, since if $\mathbb{B}$ had also corrupted Bob semi-honestly, he would have learned at least as much and we can use security of π to conclude that in that case the correct result is computed and $\mathbb{B}$ learns nothing more than $f(x_A, x_B), x_B$. In particular, the view of S can be simulated perfectly based on $f(x_A, x_B), x_B$. A symmetric conclusion holds if we switch the roles of Alice and Bob.

Now, consider the case where $S \in \mathcal{A}$ and $S \in \mathcal{B}$. We can then conclude that the view of S can be simulated perfectly based on $f(x_A, x_B), x_B$ and also based on $f(x_A, x_B), x_A$. But this must mean that the distribution of this view depends only on $f(x_A, x_B)$: assume for contradiction that there existed $x_A, x_B \neq x'_B$ such that $f(x_A, x_B) = f(x_A, x'_B)$ but the distribution of the view of S given $f(x_A, x_B), x_B$ is different from the one given $f(x_A, x_B), x'_B$. Now compare the two cases where we run the protocol on inputs (x_A, x_B) respectively (x_A, x'_B). Then the simulation based on $f(x_A, x_B), x_A$ would output the same distribution in both cases, so it cannot be consistent with both the distribution resulting from $f(x_A, x_B), x_B$ and from $f(x_A, x_B), x'_B$. So we have

PROPOSITION A.1 *If protocol π is perfectly secure in our model, it is also secure in the standard single adversary sense against semi-honest corruption of a set of servers that is in both $\mathcal{A}$ and $\mathcal{B}$, except that the simulation may not in general be efficient.*

Let us now consider malicious corruption: Alice is honest and $\mathbb{B}$ is malicious and corrupts only a set $S \in \mathcal{B}$ of servers. (so Bob is also honest). Note that from Alice's point of view, the situation is indistinguishable from a case where $\mathbb{B}$ also corrupts Bob but lets him play honestly. Security of π now implies that $\mathbb{B}$ learns nothing more than $f(x_A, x'_B)$ for some well defined input x'_B that is determined by the behaviour of the malicious servers. Note that we are not guaranteed that x'_B is equal to the honest input x_B, even though Bob plays honestly. By symmetry we get that for $S \in \mathcal{A}$, $\mathbb{A}$ will learn nothing than $f(x'_A, x_B)$.

We observe that if S is in both $\mathcal{A}$ and $\mathcal{B}$, then both the honest Alice and honest Bob are guaranteed privacy: By running π, I will give away only the function evaluated in my own input and some input from the other party. But Alice and Bob are not guaranteed to agree on the result, so we do not get security in the standard single adversary sense against malicious corruption of S.

We can in fact argue that this cannot in general be achieved in our model, even if S is in both $\mathcal{A}$ and $\mathcal{B}$: Consider a case with 3 servers S_1, S_2, S_3 and let $\mathcal{A} = \{\{S_1\}, \{S_2\}\}$ and $\mathcal{B} = \{\{S_2\}, \{S_3\}\}$. This is clearly $\mathcal{R}_2$, so our model applies. Now, it is easy to see that a secure protocol π in our sense will in this case also be semi-honestly secure against single-adversary corruption of $\{Alice, S_1\}$, as well as $\{Bob, S_3\}$. So if π was also single adversary maliciously secure against corruption of $\{S_2\}$, then we would have a situation where the whole player set is covered by 2 sets that are semi-honestly corruptible and 1 set that is maliciously corruptible, while π remains secure. And where furthermore the malicious

S_2 has no inputs or outputs. This is precisely the case where the proof of Theorem 1 in [FHM99] says we cannot have general secure MPC.

B Proof of Proposition 6.1

In this section, we show that we can indeed take the linear secret sharing schemes $\mathcal{S}_0$ and $\mathcal{S}_1$ that we have used in our constructions to be ideal, i.e., the size of every share will be the size of the secret. In our constructions, the secret is a bit-string of some fixed length m. However, clearly it is enough to show the result for the case $m = 1$ since, given ideal linear secret sharing schemes for this case, we can use them to share each bit of the secret independently.

We in fact prove the following result. Let $W \subseteq \{0,1\}^\ell$ be an affine space, i.e., $W = \mathbf{b} + V$, where $\mathbf{b} = (b_1, \ldots, b_\ell) \in \mathbb{F}_2^\ell$ and V is a vector subspace of $\mathbb{F}_2^\ell$. Consider the set $\mathcal{P}_{\ell,2}$ of 2ℓ players indexed by

$$\mathcal{P}_{\ell,2} = \{(i,j) : i = 1, \ldots, \ell, j = 0, 1\}.$$

Write Γ_W the access structure on these 2ℓ players whose minimally qualified sets are

$$\{(1,w_1),(2,w_2)\ldots,(\ell,w_\ell)\} \subseteq \mathcal{P}_{\ell,2},$$

with $(w_1, w_2, \ldots, w_\ell) \in W$.

We will show in Theorem B.1 below that there is an ideal LSSS over $\mathbb{F}_2$ with Γ_W as its access structure. Note first that this indeed yields our ideal LSSS $\mathcal{S}_0$ and $\mathcal{S}_1$. The access structures Γ_0 and Γ_1 to be realized by these schemes are precisely Γ_{V_0} and Γ_{V_1}, where V_b is the set of sharings of b in another LSSS $\mathcal{S}$. By linearity of $\mathcal{S}$, V_0 is a vector space. On the other hand V_1 is an affine space that can be described as $V_0 + \mathbf{v}_1$, where $\mathbf{v}_1$ is some sharing of 1 according to $\mathcal{S}$.

THEOREM B.1 *The access structure Γ_W admits a linear ideal secret sharing scheme over $\mathbb{F}_2$.*

PROOF. Let $V^\perp$ be the orthogonal space to V, i.e.,

$$V^\perp = \{\mathbf{h} \in \mathbb{F}_2^\ell : \langle \mathbf{v}, \mathbf{h} \rangle = 0 \text{ for all } \mathbf{v} \in V\}.$$

We define the following scheme. In order to share $s \in \mathbb{F}_2$, the dealer picks uniformly at random $r_1, \ldots, r_{\ell-1} \in \mathbb{F}_2$ and let $r_\ell = s - \sum_{i=1}^{\ell-1} r_i$. He also chooses $\mathbf{h} = (h_1, h_2, \ldots, h_\ell)$ uniformly at random in $V^\perp$.

The share of participant (i,j) is then

$$s_{i,j} = r_i + (b_i - j)h_i.$$

(Note that if $V^\perp = \{\mathbf{0}\}$, then we are just defining $s_{i,j} = r_i$).

The scheme is perfect because it is linear and ideal. Now we prove that this scheme realizes Γ_W, that is, we show that its access structure Γ coincides with Γ_W.

First we show that $\Gamma_W \subseteq \Gamma$. Let $A = \{(1,w_1), \ldots, (\ell,w_\ell)\}$ for some $\mathbf{w} \in W$, which is therefore of the form $\mathbf{w} = \mathbf{v} + \mathbf{b}$ with $\mathbf{v} \in V$. The sum of all shares of participants in A is:

$$\sum_{(i,j)\in A} s_{i,j} = \sum_{i=1}^{\ell} (r_i + (b_i - w_i)h_i) = \sum_{i=1}^{\ell} r_i - \langle \mathbf{v}, \mathbf{h} \rangle = \sum_{i=1}^{\ell} r_i = s,$$

which shows the claim.

We now show that $\Gamma \subseteq \Gamma_W$. For this we prove that: 1) every minimally qualified set in Γ has to contain exactly one player (i, y_i) for every i, and 2) if $(y_1, \ldots, y_\ell) \notin W$, then $\{(1,y_1),(2,y_2),\ldots,(\ell,y_\ell)\}$ is not in Γ. This will show that all minimally qualified sets are in Γ_W and hence $\Gamma \subseteq \Gamma_W$.

Part 1 is shown as follows. Obviously, every qualified set has to contain at least one player (i, y_i) for each i, since all r_i's are needed to reconstruct s. On the other hand, since the scheme is linear, a set of players A is qualified if there exists a linear function ρ_A such that s can be recovered by applying ρ_A to the shares in A, for every possible secret and choice of the randomness. But since we are working over the field $\mathbb{F}_2$ this just means that $A \in \Gamma$ if and only if there exists $A' \subseteq A$, such that $s = \sum_{(i,j)\in A'} s_{i,j}$. This means A is minimally qualified if and only if it coincides with this A', i.e., the secret can be recovered as the sum of the shares in A. Now if both $(i, 0)$ and $(i, 1)$ are in A, r_i appears twice in the sum and will cancel out. Hence it can only be the case that A contains exactly one element of the form (i, y_i) for each i.

Now for the second part, we show that if $A = \{(1, y_1), (2, y_2), \ldots, (\ell, y_\ell)\}$ and $(y_1, \ldots, y_\ell) \notin W$ then there is a valid sharing of $s = 1$ where every player in A has 0 as share. Since this is also the case for $s = 0$, this will show $A \notin \Gamma$. If $\mathbf{y} = (y_1, \ldots, y_\ell) \notin W$, that means $\mathbf{y} - \mathbf{b} \notin V$ and there exists some $\mathbf{h}' \in V^\perp$ such that $\langle \mathbf{y} - \mathbf{b}, \mathbf{h}' \rangle = 1$. Now consider the following choice of the randomness: $\mathbf{h} = \mathbf{h}'$ and $r_i = (y_i - b_i)h_i$ for $i = 1, \ldots, \ell - 1$. Then if $s = 1$ is shared using this randomness, it is immediate that $s_{i,y_i} = 0$ for every $i = 1, \ldots, \ell - 1$, and we easily verify that

$$s_{\ell,y_\ell} = (s - \sum_{i=1}^{\ell-1} r_i) + (b_\ell - y_\ell)h_\ell = 1 - \langle \mathbf{y} - \mathbf{b}, \mathbf{h} \rangle = 1 - 1 = 0$$

which finalizes the proof. △

C Protocol for general $\mathcal{A}$

We show the general version of the protocol π_{OT} from Section 6, when the adversary structure $\mathcal{A}$ corrupted by $\mathbb{A}$ is not necessarily the adversary structure of an ideal LSSS over $\mathbb{F}_2$. Note that many interesting access structures, for example most threshold structures, do not admit an ideal LSSS over $\mathbb{F}_2$.

Let $\mathcal{S}$ be a possibly non-ideal perfect secret sharing scheme with adversary structure $\mathcal{A}$. For $i = 1, \ldots, n$ the i-th share of $\mathcal{S}$ belongs to some vector space $U_i = \mathbb{F}_2^{\ell_i}$ for some integer $\ell_i \geq 1$. Let $\ell = \sum_{i=1}^n \ell_i$ be the complexity of $\mathcal{S}$.

The idea of the generalization is simple. Basically S_i is splitted in ℓ_i subservers, each of which receives one bit from Bob and two shares of each m_0 and m_1 from Alice and performs an OT as servers did in the protocol from Section 6 (we remark however that the adversaries corrupt full servers and not individual subservers).

More precisely, let $V_0, V_1 \subseteq U_1 \times \cdots \times U_n$ the sets of all possible sharings of 0 and 1 respectively. We can think of the elements of V_0 and V_1 as k-bit strings, and we index their coordinates by pairs (i, k) where the (i, k)-th coordinate of a sharing is the k-th bit of the i-th share.

As before we construct two perfect secret sharing schemes that we call $\mathcal{S}_0$ and $\mathcal{S}_1$. These are now secret sharing schemes with 2ℓ shares each and the set of shares will be indexed by

$$\mathcal{P}_{\ell,2} := \{(i, k, j) : i = 1, \ldots, n,\ k = 1, \ldots, \ell_i,\ j = 0, 1\}.$$

We define the access structure Γ_0 of $\mathcal{S}_0$ as follows. The minimally qualified sets are exactly the sets

$$\{(1, 1, v_{(1,1)}), (1, 2, v_{(1,2)}) \ldots, (n, k_n, v_{(n,k_n)})\} \subseteq \mathcal{P}_{\ell,2},$$

with $(v_{(1,1)}, v_{(1,2)}, \ldots, v_{(n,k_n)}) \in V_0$. Similarly, the access structure Γ_1 of $\mathcal{S}_1$ has as its family of minimally qualified sets

$$\{(1, 1, v_{(1,1)}), (1, 2, v_{(1,2)}) \ldots, (n, k_n, v_{(n,k_n)})\} \subseteq \mathcal{P}_{\ell,2},$$

with $(v_{(1,1)}, v_{(1,2)}, \ldots, v_{(n,k_n)}) \in V_1$.

Again, we can construct ideal schemes $\mathcal{S}_0, \mathcal{S}_1$ with the properties above from Theorem B.1.

The general protocol is given in Figure 7. The security proofs work essentially as in the particular case presented in Sections 6 and 7.2.

Oblivious transfer protocol π_{OT} (non-ideal $\mathcal{S}$ case)

1. Alice independently creates sharings

$$[m_0]_{\mathcal{S}_0} = (m_0^{(i,k_i,j)})_{i\in\{1,\ldots,n\},k_i\in\{1,\ldots,\ell_i\},j\in\{0,1\}}$$

and

$$[m_1]_{\mathcal{S}_1} = (m_1^{(i,k_i,j)})_{i\in\{1,\ldots,n\},k_i\in\{1,\ldots,\ell_i\},j\in\{0,1\}}$$

for her inputs.
Bob creates a sharing $[b]_{\mathcal{S}} = (b_{(i,k_i)})_{i\in\{1,\ldots,n\},k_i\in\{1,\ldots,\ell_i\}}$ of his input, where each $b_{(i,k_i)} \in \{0,1\}$.

2. Bob sends $(\texttt{Bob-share}, i, b_i)$ to server S_i, where $b_i = (b_{(i,1)}, \ldots, b_{(i,\ell_i)})$.

3. S_i sends $(\texttt{ready}, i)$ to Alice.

4. After Alice has received $(\texttt{ready}, i)$ from every server S_i, she sends the messages $(\texttt{Alice-share}, i, (u_i^{k_i,j})_{k_i\in\{1,\ldots,\ell_i\},j\in\{0,1\}})$ to each server S_i where $u_i^{k_i,j} := m_0^{(i,k_i,j)}||m_1^{(i,k_i,j)}$.

5. Server S_i sends $(\texttt{output}, i, (u_i^{k_i,b_{(i,k_i)}})_{k_i\in\{1,\ldots,\ell_i\}})$ to Bob.

6. If for any i, $u_i^{k_i,b_{(i,k_i)}} \notin \mathcal{M}^2$, then Bob outputs $\mathbf{0}$. Otherwise, Bob parses each $u_i^{k_i,b_{(i,k_i)}}$ as $m_0^{(i,k_i,b_{(i,k_i)})}||m_1^{(i,k_i,b_{(i,k_i)})}$.
If $b = 0$, Bob constructs m_0 by applying

$$\texttt{Reconstruct}_{\mathcal{S}_0}(\{((i,k_i,b_{(i,k_i)}), m_0^{(i,k_i,b_{(i,k_i)})}) : i \in \mathcal{P}_n, k_i \in \{1,\ldots,\ell_i\}).$$

If $b = 1$, Bob constructs m_1 by applying

$$\texttt{Reconstruct}_{\mathcal{S}_1}(\{((i,k_i,b_{(i,k_i)}), m_1^{(i,k_i,b_{(i,k_i)})}) : i \in \mathcal{P}_n, k_i \in \{1,\ldots,\ell_i\}).$$

In any of the cases, if the reconstruction fails, output $\mathbf{0}$. Otherwise output the reconstructed m_b.

Figure 7: Protocol π_{OT}

D Description of $\mathcal{F}_{OT}^L$ by cases

We describe the functionality $\mathcal{F}_{OT}^L$ (the composition of $\mathcal{I}_{\text{HHA}}$, $\mathcal{F}_{CIOT}$ and $\mathcal{I}_{\text{HHB}}$) case-by-case, according to the level of corruption of Alice and Bob.

In every case, the functionality receives $(\texttt{corrupt}, c_A, A)$ and $(\texttt{corrupt}, c_B, B)$ from the local simulators.

1. **Case HH** ($c_A = c_B =$ Honest).
It works exactly as $\mathcal{F}_{OT}$.

2. **Case HS** ($c_A =$ Honest, $c_B =$ Semi-honest).
It awaits $(\texttt{Bob-share}, i, b_i)$ for all $i \in \mathcal{P}_n$.
On input $(\texttt{transfer}, b)$ from Sim_B, it sends $(\texttt{ready})$ to Sim_A.
On input $(\texttt{send}, m_0, m_1)$ from Sim_A the functionality generates a random message $m'_{1-b} \in \mathcal{M}$ and random sharings $[m_b]_{\mathcal{S}_b}$ and $[m'_{1-b}]_{\mathcal{S}_{1-b}}$. It creates values $u_i^j = m_0^{(i,j)}||m_1'^{(i,j)}$ (if $b = 0$) or $u_i^j = m_0'^{(i,j)}||m_1^{(i,j)}$ (if $b = 1$).

It sends, to Sim_B, the messages $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ for $i \in B$ and $(\texttt{output}, i, u_i^{b_i})$ for $i \notin B$.

3. **Case SH** (c_A = Semi-honest, c_B = Honest).
   On input $(\texttt{transfer}, b)$ from Sim_B, it generates shares b'_i for a random bit b' and sends to Sim_A the messages $(\texttt{Bob-share}, i, b'_i)$ for all $i \in A$ and $(\texttt{ready}, i)$ for all $i \notin A$.

   On input $(\texttt{send}, m_0, m_1)$ from Sim_A, it sends $(\texttt{sent}, m_b)$ to Sim_B.

4. **Case SS** (c_A = Semi-honest, c_B = Semi-honest).
   The functionality awaits, for all $i \in \mathcal{P}_n$, the messages $(\texttt{Bob-share}, i, b_i)$ from Sim_B.

   It sends to Sim_A the messages $(\texttt{Bob-share}, i, b_i)$ for all $i \in A$ and $(\texttt{ready}, i)$ for all $i \notin A$.

   The functionality awaits, for all $i \in \mathcal{P}_n$, the messages $(\texttt{Alice-share}, i, u_i^0, u_i^1)$.

   It sends, to Sim_B, the messages $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ for $i \in B$ and $(\texttt{output}, i, u_i^{b_i})$ for $i \notin B$.

5. **Case HM** (c_A = Honest, c_B = Malicious).
   The functionality acts exactly the same as in the HS case.

6. **Case MH** (c_A = Malicious, c_B = Honest).
   The functionality acts exactly the same as in the SH case.

7. **Case SM** (c_A = Semi-honest, c_B = Malicious).
   The functionality acts the same as in the SS case except that *all* messages received from Sim_B are sent to Sim_A.

8. **Case MS** (c_A = Malicious, c_B = Semi-honest).
   The functionality acts the same as in the SS case except that *all* messages received from Sim_A are sent to Sim_B.

E Proof of Theorem 7.4

We first describe how each of the local simulators works. Later on, we will show the indistinguishability between the real world and the ideal world with these simulators.

E.1 Simulators

E.1.1 Description of Sim_A

- First, the simulator awaits ($\texttt{corrupt}$, c_A, A) and forwards it to $\mathcal{I}_{\mathrm{HHA}}$. It also takes note of this tuple.

- If c_A = Honest, it awaits ($\texttt{ready}$) from the functionality and forwards it to the environment. It then awaits $(\texttt{send}, m_0, m_1)$ from the environment and sends it to $\mathcal{I}_{\mathrm{HHA}}$ and ignores any other message.

- If c_A = Semi-honest, on receiving a share b_i or a message ($\texttt{ready}$, i) from $\mathcal{I}_{\mathrm{HHA}}$, it forwards them to the environment. It also forwards any other message from $\mathcal{I}_{\mathrm{HHA}}$ that contains an index i with $i \in A$ to the environment[4]. It then awaits the message $(\texttt{send}, m_0, m_1)$ from the environment. It sends the message ($\texttt{send}$, m_0, m_1) to $\mathcal{I}_{\mathrm{HHA}}$. It then generates the values $\{(u_i^0, u_i^1)\}$ from

[4] Note this captures the situation where a malicious $\mathbb{B}$ sends arbitrary messages to servers with $i \in A \cap B$, since $\mathbb{A}$ will see those messages in the real protocol.

(m_0, m_1) as in the protocol. It sends these values to the environment and it sends the messages $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ to $\mathcal{I}_{\mathrm{HHA}}$ for all i.

- If c_A = Malicious, during its whole interaction with the environment, on reception of messages from the environment it checks that they contain a unique index i corresponding to a server. For each message, if this does not happen or if $i \notin A$ (unless for messages of the form $(\texttt{Alice-share}, i, u_i^0, u_i^1)$), the message is ignored. Otherwise, it is forwarded to $\mathcal{I}_{\mathrm{HHA}}$[5]. On reception of messages from $\mathcal{I}_{\mathrm{HHA}}$, it forwards them to the environment.
  On reception of the shares of b_i for $i \in A$ from $\mathcal{I}_{\mathrm{HHA}}$, it also constructs sharings $[0]_{\mathcal{S}} := (c_1, \ldots, c_n)$, $[1]_{\mathcal{S}} := (d_1, \ldots, d_n)$ consistent with the received b_i's (i.e., $c_i = d_i = b_i$ for $i \in A$). On reception of the values $\{u_i^j : (i,j) \in \mathcal{P}_{n,2}\}$ from the environment, it also constructs

$$m_0 = \texttt{Reconstruct}_{\mathcal{S}_0}(\{((i, c_i), m_0^{(i,c_i)}) : i \in \mathcal{P}_n\})$$

  and

$$m_1 = \texttt{Reconstruct}_{\mathcal{S}_1}(\{((i, d_i), m_1^{(i,d_i)}) : i \in \mathcal{P}_n\}).$$

  If the reconstruction of m_0 (respectively m_1) fails, Sim sets $m_0 = \mathbf{0}$ (resp. $m_1 = \mathbf{0}$). Now it sends the command (send, m_0, m_1) to $\mathcal{I}_{\mathrm{HHA}}$.

  Note that in the case that Bob is honest, the local simulator Sim_B for Bob will output the value that was generated by Sim_A, otherwise Sim_B will reconstruct a message based on the shares received via $\mathcal{F}_{OT}^L$.

E.1.2 Description of Sim_B

- First, the simulator awaits (corrupt, c_B, B), notes that value and forwards it to $\mathcal{I}_{\mathrm{HHB}}$.
- If c_B = Honest, it sends (transfer, b) to $\mathcal{I}_{\mathrm{HHB}}$ and forwards the response $(\texttt{sent}, m_b)$ to the environment.
- If c_B = Semi-honest, it awaits input (transfer, b). It forwards it to $\mathcal{I}_{\mathrm{HHB}}$. It then selects a random sharing $[b]_{\mathcal{S}} = (b_i)_{i \in \mathcal{P}_n}$. It sends the b_i to the environment. It sends messages $(\texttt{Bob-share}, i, b_i)$ to $\mathcal{I}_{\mathrm{HHB}}$ for every i. On receiving $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ for $i \in B$ and $(\texttt{output}, i, u_i^{b_i})$ for $i \notin B$ from $\mathcal{I}_{\mathrm{HHB}}$ it forwards these to the environment. It reconstructs a message m_b from the received shares as in the protocol. Finally it forwards this message to the environment.
- If c_B = Malicious, during the whole interaction with the environment, on reception of messages from the environment it checks that they contain a unique index i corresponding to a server. For each message, if this does not happen or if $i \notin B$ (unless for messages of the form $(\texttt{Bob-share}, i, b_i)$), the message is ignored. Otherwise, it is forwarded to $\mathcal{I}_{\mathrm{HHB}}$. On reception of messages from $\mathcal{I}_{\mathrm{HHA}}$, it forwards them to the environment. The simulator awaits that the environment has sent $(\texttt{Bob-share}, i, b_i)$ to each $i \in \overline{B}$. On receiving $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ for $i \in B$ and $(\texttt{output}, i, u_i^{b_i})$ for $i \notin B$ from $\mathcal{I}_{\mathrm{HHB}}$ it forwards these to the environment. It reconstructs a message m_b from the received shares as in the protocol. Finally it forwards this message to the environment.

[5] This captures the fact that in the real protocol, a malicious $\mathbb{A}$ can only deviate from the protocol by interacting with a server S_i either arbitrarily, in the case $i \in A$, or by sending messages to them, in the case $i \notin A$; however, in the latter case all messages which are not of the form $(\texttt{Alice-share}, i, u_i^0, u_i^1)$ will be ignored.

E.2 Indistinguishability

First we argue that the case where one of the players is honest reduces to the universal composability proof. Say Alice is honest. Then, the idea is that the functionality $\mathcal{F}_{CIOT}$ (the "inner" part of $\mathcal{F}_{OT}^{L}$) is basically the same as the functionality $\mathcal{F}_{OT}$ in Section 6, except that it sends some additional messages to the honest side. However, these messages are ignored by $\mathcal{I}_{\mathrm{HHA}}$. Moreover, the composition of Sim_B and $\mathcal{I}_{\mathrm{HHB}}$ acts as the simulator for the UC proof in the case of an honest Alice. If Bob is honest, the same holds by swapping A and B.

As for the cases where both Alice and Bob are corrupted (respectively by $\mathbb{A}$ and $\mathbb{B}$), as we have said, we are assuming that $\mathbb{A}$ and $\mathbb{B}$ are not both malicious, and thus one of them is semi-honest. Say for the moment that $\mathbb{A}$ is semi-honest. If we compare this with the situation where Alice is honest, and Bob has the same level of corruption, here we need to take into account that, in the real world, $\mathcal{Z}$ will additionally receive from $\mathbb{A}$ the information of the servers S_i with $i \in A$, and all information held by Alice, and all this information is consistent with what it receives from $\mathbb{B}$. We need to show that this needs to be the case also in the simulation. However, note that by design, the ideal functionality transfers the appropiate parts of the sharings created by Sim_B to Sim_A. Moreover, if $\mathbb{B}$ is malicious it also sends any other potential information that goes through the servers corrupted by $\mathbb{A}$.

The environment then receives from each of the simulators the sharings created by themselves as well as the shares received from the other simulator via the functionality. This implies that the information received by $\mathcal{Z}$ in both sides is also consistent in the ideal world. Moreover, it is indistinguishable from the view in the real world. This follows by the same arguments as above. Again, the case of a semi-honest Bob (and corrupted Alice) is analogous.