

Network Oblivious Transfer

Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon

MIT

Abstract. Motivated by the goal of improving the concrete efficiency of secure multiparty computation (MPC), we study the possibility of implementing an infrastructure for MPC. We propose an infrastructure based on oblivious transfer (OT), which would consist of OT channels between some pairs of parties in the network. We devise information-theoretically secure protocols that allow additional pairs of parties to establish secure OT correlations using the help of other parties in the network in the presence of a dishonest majority. Our main technical contribution is an upper bound that matches a lower bound of Harnik, Ishai, and Kushilevitz (Crypto 2007), who studied the number of OT channels necessary and sufficient for MPC. In particular, we characterize which n -party OT graphs G allow t -secure computation of OT correlations between all pairs of parties, showing that this is possible if and only if the complement of G does not contain the complete bipartite graph $K_{n-t, n-t}$ as a subgraph.

Keywords: Secure multiparty computation, oblivious transfer, infrastructure.

1 Introduction

Protocols for secure multiparty computation [65,31,8,16] allow a set of mutually distrusting parties to carry out a distributed computation without compromising the privacy of inputs or the correctness of the end result. As a research area, secure computation has witnessed several breakthroughs in the last decade [66,47,43,59,54,53,40,57,41,52]. However, despite a wide array of potential game-changing applications, there is nearly no practical adoption of secure computation today (with the notable exceptions of [11,12]). Computations wrapped in a secure computation protocol do not yet deliver results efficiently enough to be acceptable in many cloud-computing applications. For instance, state-of-the-art semihonest 2-party protocols incur a factor ≈ 100 slowdown even for simple computations. As we move into the world of wearables, such a slowdown would effectively close down many important avenues for application of secure computation.

In the absence of practical real-world protocols for secure computation which are secure in the presence of any number of dishonest parties, there is a need for relaxations that are meaningful and yet provide significant performance benefits. As an example, classic protocols for secure computation [8,16,62] (with subsequent improvements e.g., [19,9,4,23,21,20]) offer vastly better efficiency at the cost of tolerating only a small constant fraction of adversaries. The resilience

offered is certainly acceptable when the number of participating parties is large, e.g., the setting of *large-scale* secure computation [13,25,67,14]. Although large-scale secure computation is well-suited for several interesting applications (such as voting, census, surveys), we posit that typical settings involve computations over data supplied by a few end users. In such cases, the overhead associated with interaction among a large number of *helper parties* is likely to render these protocols more expensive than a standard secure computation protocol among the end users. If the number of helper parties is small, security against a small fraction of corrupt parties may be a very weak guarantee, since a handful of corrupt parties could render the protocol insecure.

An orthogonal approach for reducing the online cost of secure computation protocols is the use of *preprocessing* [3,24,10,1]. This approach can dramatically reduce the cost of secure computation: for instance, given preprocessing [3], the ≈ 100 factor slowdown for simple computations no longer applies. Recent theoretical research has shown that many primitives can even be made *reusable* (e.g. [34]). Perhaps the most important drawback of this approach (other than the fact that the preprocessing phase is typically very expensive) is that the preprocessing is not *transferable*. Clearly, a pair of parties that want to perform a secure computation cannot benefit from this approach without performing the expensive preprocessing step. Moreover, this seems to hold even if each of the two parties have set up the preprocessing with multiple others. Typically, the cost of the preprocessing phase is quite high, presenting a barrier for the practical use of preprocessed protocols. This is especially true in settings where parties are unlikely to run many secure computations which would amortize the cost of preprocessing.

Motivated by the discussion above, we conclude that some directions that seem to offer efficiency benefits for secure computation are (1) highly resilient protocols that use only a small number of helper parties, and (2) a preprocessing procedure that allows a notion of transferability between users. Taken together, these two ideas have the potential to provide an *infrastructure* for efficient secure computation. Some sets of parties might run a preprocessing phase among themselves. These parties can then act as helper parties and “transfer” their preprocessing to help users who want to run a secure computation protocol. We informally describe some desiderata for such an infrastructure:

- *Reusability/Amortization*. Setting up an infrastructure component could be expensive, but using it and maintaining it should be inexpensive relative to setting up a new component.
- *Transferability/Routing*. It should be possible to combine different components of the infrastructure to deliver benefits to the end users.
- *Robustness/Fault-tolerance*. Failure or unavailability of some components of the infrastructure should not nullify the usefulness of the infrastructure.

It is not hard to see that the above criteria are fulfilled for infrastructures that we use in daily life, for e.g., the infrastructure for online communication (e-mail, instant messaging, etc.) consisting of transatlantic undersea cables, routers, wireless access points, etc. What cryptographic primitives would be good can-

didates for a *secure computation infrastructure*? In this work, we explore the possibility of using *oblivious transfer* [61,27] for this purpose.

1.1 Our Model: Network Oblivious Transfer

Oblivious transfer (OT) is a fundamental building block of secure computation [46,45]. As discussed in [45], some of the benefits of basing secure computation on OT include:

- *Preprocessing.* OT enables precomputation in an offline stage before the inputs or the function to be computed are known. The subsequent online phase is extremely efficient [3].
- *Amortization.* The cost of computing OTs can be accelerated using efficient OT extension techniques [2,43,45,59].
- *Security.* OTs can be realized under a wide variety of computational assumptions [60,27,61,58,18] or under physical assumptions.

In this work, we consider n parties connected by a synchronous network with secure point-to-point private communication channels between every pair of parties. In addition, some pairs of parties on the network have established *OT channels* between them providing them with the ability to perform arbitrarily many OT operations. We represent the OT channel network via an *OT graph* G . The vertices of G represent the n parties, and pairs of parties that have an established OT channel are connected by an edge in G . Since OT can be reversed unconditionally [63], we make no distinction between the sender and the receiver in an OT channel. This OT graph represents the infrastructure we begin with. The OT channels could either represent $\text{poly}(\lambda)$ 1-out-of-2 OT correlations for a computational security parameter λ , or a physical channel (e.g., noisy channel) that realizes, say δ -Rabin OT [61].¹ We are interested in obtaining security against adaptive semihonest adversaries. We also discuss security against adaptive malicious adversaries under computational assumptions.

Two parties that are connected by an edge can use the corresponding existing OT channel to run a secure computation protocol between themselves. What about parties that are not connected by an edge? Clearly, they can establish an OT channel between themselves via an OT protocol [60,18] or perhaps using a physical channel. The latter option, if possible, is likely to be expensive and the costs of setting up a physical channel may be infeasible unless the two parties are likely to execute many secure computation protocols. The former option is also expensive as it involves use of public-key cryptography which is somewhat necessary in the light of [42].² This motivates the question of whether additional parties can use an existing OT infrastructure to establish an OT

¹ Recall that λ 1-out-of-2 OT correlations can be extended to $\text{poly}(\lambda)$ 1-out-of-2 OT correlations via OT extension using just symmetric-key cryptography (e.g. one-way functions [2] or correlation-robust hash functions [43]).

² As a rule of thumb, use of public-key cryptography is computationally around 4-6 orders of magnitude more expensive than using symmetric-key cryptography [7].

channel between themselves unconditionally or relying only on the existence of symmetric-key cryptography. A positive result to this question would show that expensive cryptographic operations are not required to set up additional OT channels which could be used for efficient secure computation. In this work we construct OT protocols with information-theoretic security against a threshold adversary. See Appendix A for more discussion.

1.2 Related Work and Our Contributions

Related work. As mentioned previously, there is a large body of work on secure computation in the offline/online model (cf. [51,50,24,10,59] and references therein). These protocols exhibit an extremely fast online phase at the expense of a slow preprocessing phase (sometimes using MPC [51] or more typically, OT correlations [59] or a somewhat homomorphic encryption scheme [24]). To the best of our knowledge, the question of *transferability* of preprocessing has not been explicitly investigated in the literature with the notable exception of [36], which we will discuss in greater detail below. There is a large body of work on secure computation against a threshold adversary (e.g. [8,16,62,31]). Popular regimes where secure computation against threshold adversaries have been investigated are for $t < n/3$, $t < n/2$, or $t = n - 1$. In this work we are interested in threshold adversaries for a dishonest majority, that is, adversaries which can corrupt t out of n parties for $n/2 \leq t < n$.³ Such regimes were investigated in other contexts such as authenticated broadcast [29] and fairness in secure computation [6,39,44]. Infrastructures for *perfectly secure message transmission* (PSMT) were investigated in the seminal work of [26] (see also [28] and references therein). While the task of PSMT is similar to our question regarding OT channels, there are inherent differences. For example, our protocols can implement OT even between two parties that are isolated in the OT graph (i.e., not connected to any other party via an OT channel).⁴ In PSMT, on the other hand, there is no hope of achieving secure communication with a node that is not connected by any secure channel.

Most relevant to our results is the work of Harnik, Ishai, and Kushilevitz [36]. The main question in their work is an investigation of the number of OT channels sufficient to implement a n -party secure computation protocol. In a nutshell, they show against an adaptive t -threshold adversary for $t = (1 - \delta)n$, an explicit construction of an OT graph consisting of $(n + o(n))\binom{\lceil 1/\delta \rceil}{2}$ OT channels that suffices to implement secure computation among the n parties. They note further that against a static adversary, $\binom{\lceil s/\delta \rceil}{2}$ OT channels suffice, where s denotes a statistical security parameter. On the negative side, they show that a complete OT graph is necessary for secure computation when dealing with an adversary that can corrupt $t = n - 1$ parties. They derive this result by showing that in a

³ When $t < n/2$, there is no need to rely on an OT infrastructure [62].

⁴ Recall that the model considered in this work, we assume a *full* network of secure private communication channels.

3-party OT graph with two OT channels, it is not possible to obtain OT correlations between the third pair of parties with security against two corruptions. Moreover they generalize their 3-party negative result to any OT graph whose complement contains the complete bipartite graph $K_{n-t, n-t}$ as a subgraph. In our paper we extend and generalize the results of [36], characterizing the networks for which it is possible to obtain OT correlations between a designated pair of parties. We now proceed to explain our contributions in more detail.

Our contributions. We introduce our main result:

Theorem (informal). Let G be an OT graph on n parties P_1, \dots, P_n , so that any pair of parties P_i, P_j which are connected by an edge may make an unbounded number of calls to an OT oracle. Let \mathbb{A} be the class of semihonest t -threshold adversaries which may adaptively corrupt at most t parties.⁵ Then two parties A and B with $A, B \in \{P_1, \dots, P_n\}$ can information-theoretically emulate an OT oracle while being secure against all adversaries $\mathcal{A} \in \mathbb{A}$ if and only if

1. (honest majority) it holds that $t < n/2$; or
2. (trivial) A and B are connected by an edge in G ; or
3. (partition) there exists no partition V_1, V_2, V of G such that all of the following conditions are satisfied: (a) $|V_1| = |V_2| = n - t$ and $|V| = 2t - n$; (b) $A \in V_1$ and $B \in V_2$; and (c) for every $A' \in V_1$ and $B' \in V_2$ it holds that $(A', B') \notin G$.

Our main theorem gives a complete characterization of networks for which a pair of parties can utilize the OT network infrastructure to execute a secure computation protocol. The first two conditions in our theorem are straightforward: (1) if $t < n/2$, then we are in the honest majority regime, and thus it is possible to implement secure computation (or emulate an OT oracle) using the honest majority information-theoretically secure protocols of [62]; (2) clearly if A and B are connected by an OT edge then by definition they can emulate an OT oracle.

Condition (3) applies when $t \geq n/2$ and when A and B do not have an OT edge between them. This condition is effectively the converse of the impossibility result of [36], which states that any n -party OT graph whose complement contains $K_{n-t, n-t}$ as a subgraph cannot allow a n -party secure computation that tolerates t semihonest corruptions. Condition (3) implies that any n -party OT graph whose complement does not contain $K_{n-t, n-t}$ as a subgraph can run n -party secure computations tolerating t semihonest corruptions.

Applying our main theorem. We first compare our positive results to those of [36]. They investigate how to construct an OT graph with the minimum number of edges allowing n parties to execute a secure computation protocol. They show a construction for a graph with $(n + o(n))\binom{1/\delta}{2}$ edges which they prove is sufficient for resilience against an adversary that corrupts $(1 - \delta)n$ parties. Our

⁵ Combining our work with results from [35], we can also obtain computational security against nonadaptive malicious adversaries.

result provides a complete, simple characterization of which OT graphs on n vertices are sufficient to run a t -secure protocol generating OT correlations between all pairs of vertices for any $t \geq n/2$, which is sufficient to obtain a protocol for secure computation among the n parties [46,45]. Our main theorem also implies that determining the minimum number of OT edges needed to execute a secure computation protocol for general $n, t \geq n/2$ is equivalent to an open problem in graph theory posed by Zarankiewicz in 1951 [48].

Our results immediately imply that for some values of t , extremely simple sparse OT graphs suffice for achieving secure multiparty computation. For n even and $t = n/2$, we have that the t -claw graph (cf. Fig. 4(a)) has t edges and suffices to achieve t -secure multiparty computation. For n odd and $t = (n+1)/2$, the $(t+1)$ -cycle has $t+1$ edges and suffices to achieve t -secure multiparty computation. We show in Appendix F that these examples are the sparsest possible graphs which can achieve $\lfloor (n+1)/2 \rfloor$ -secure multiparty computation.

Next, our results are also well-suited to make use of an OT infrastructure for secure computation. Specifically, let G_I denote the OT graph consisting of existing OT edges between parties that are part of the infrastructure. Now suppose a pair of parties A, B not connected by an OT edge wish to execute a secure computation protocol. Then they can find a subgraph G of G_I with $A, B \in G$ and $|G| = n$ such that they agree that at most t out of the n parties can be corrupt and the partition condition in our main theorem holds for G . Since it is possible to handle a dishonest majority, parties do not have to settle for a lower threshold and can enjoy increased confidence in the security of their protocol by making use of the infrastructure. Surprisingly, it turns out the OT subgraph G need not even contain t OT edges to offer resilience against t corruptions (cf. Fig. 2(c) with $n = 4, t = 2$).

A pair of parties may use the OT correlations generated as the base OTs for an OT extension protocol and inexpensively generate many OT correlations that can be saved for future use or to add to the OT infrastructure. In any case, it should be clear that our protocols readily allow load-balancing across the OT infrastructure and are also abort-tolerant in the sense that if some subgraph G ends up not delivering the output, then one can readily use a different subgraph G' . Thus we believe that our results can be used to build a *scalable* infrastructure for secure computation that allows (1) amortization, (2) routing, and (3) is robust.

An important caveat regarding efficiency. In the special cases $t = n/2 + \mathcal{O}(1)$ and $t = n - \mathcal{O}(1)$, determining whether a graph satisfies the partition condition requires at most $\text{poly}(n)$ time. However, in general the problem is coNP-complete, since it can be restated in the graph complement as subgraph isomorphism of a complete bipartite graph [30]. Our protocols are efficient in n only for $t = n/2 + \mathcal{O}(1)$ and $t = n - \mathcal{O}(1)$.⁶ Computing OT correlations in the presence

⁶ For $t = n/2 + \mathcal{O}(1)$, we achieve efficiency using computationally-secure OT extension (e.g. [2,43]). Our protocol with information-theoretic security is quasipolynomial-time for $t = n/2 + \mathcal{O}(1)$. We do, however, achieve information-theoretic security in polynomial time for $t = n - \mathcal{O}(1)$.

of a dishonest majority may, in particular, be useful in practice for small values of n , for which our protocol is very efficient.

Organization After discussing preliminaries in Section 2, we give an overview of some of our techniques in Section 3, where we show solutions for the simple case when $n = 4$ and $t = 2$. Following this we briefly sketch the lower bound in Section 4 and describe the building blocks required for our upper bounds in Section 5. The rest of the paper is devoted to proving the upper bound first for the specific cases of $t = n/2$ (Section 6) and $t = n - 2$ (Section 7). We then use each of these protocols in two different solutions for the general case of $t \geq n/2$ in Section 8 which are efficient for different values of t .

2 Preliminaries

2.1 Notation and definitions

Let \mathcal{X}, \mathcal{Y} be two probability distributions over some set S . Their *statistical distance* is

$$\mathbf{SD}(\mathcal{X}, \mathcal{Y}) \stackrel{\text{def}}{=} \max_{T \subseteq S} \{\Pr[\mathcal{X} \in T] - \Pr[\mathcal{Y} \in T]\}$$

We say that \mathcal{X} and \mathcal{Y} are ϵ -close if $\mathbf{SD}(\mathcal{X}, \mathcal{Y}) \leq \epsilon$ and this is denoted by $\mathcal{X} \approx_\epsilon \mathcal{Y}$. We say that \mathcal{X} and \mathcal{Y} are identical if $\mathbf{SD}(\mathcal{X}, \mathcal{Y}) = 0$ and this is denoted by $\mathcal{X} \equiv \mathcal{Y}$.

All graphs addressed in this work are undirected. We denote a graph as $G = (V, E)$ where V is a set of vertices and E is a set of edges. We denote an edge e as $e = \{v_1, v_2\}$, where $v_1, v_2 \in V$.

Let K_n denote the complete graph on n vertices for any $n \in \mathbb{N}$. Let A_a^s denote the graph $G = (V, E)$ on $2a + s$ vertices with $V = V_A \cup V_S \cup V_B$, where $|V_A| = |V_B| = a$ and $|V_S| = s$, and

$$E = \{\{v_1, v_2\} : v_1 \notin V_A \vee v_2 \notin V_B\}$$

We will sometimes consider subgraphs of A_a^s which preserve labels of vertices. In this case we will always label the vertices so that vertex $A \in V_A$ and vertex $B \in V_B$.

For two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same vertex set V , we say that G_1 and G_2 are (v_1, \dots, v_ℓ) -*isomorphic*, denoted by $G_1 \simeq_{v_1, \dots, v_\ell} G_2$, if the two graphs are isomorphic to one another while fixing the labelings of vertices $v_1, \dots, v_\ell \in V$, that is, there exists an isomorphism σ such that $\sigma(v_i) = v_i$ for all $i \in [\ell]$.

Similarly, given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \subseteq V_2$ and $v_1, \dots, v_\ell \in V_1$, we say that G_1 is a (v_1, \dots, v_ℓ) -*subgraph* of G_2 , denoted $G_1 \subseteq_{v_1, \dots, v_\ell} G_2$, if G_1 is (v_1, \dots, v_ℓ) -isomorphic to some subgraph of G_2 .

In particular, in the special case that graph $G = (V, E)$ contains vertices $A, B \in V$, we say that G is an (A, B) -*subgraph* of A_a^s (or that $G \subseteq_{A, B} A_a^s$) if there is an isomorphism σ between G and a subgraph of A_a^s such that A is

mapped into set V_A and B is mapped into set V_B (that is, $\sigma(A) \in V_A$ and $\sigma(B) \in V_B$).

Call an n -vertex graph $G = (V, E)$ k -*unsplittable* for $k \leq n/2$ if any two disjoint sets of k vertices have some edge between them. That is, G is k -unsplittable if for all partitions of the vertices V into three disjoint sets V_1, V_2, V_3 of sizes $|V_1| = |V_2| = k$ and $|V_3| = n - 2k$, there exists some edge $(u, v) \in E$ with $u \in V_1, v \in V_2$. It is immediate from this definition that G is k -unsplittable if and only if $G \not\subseteq \Lambda_k^{n-2k}$.

Similarly, call G (k, A, B) -*unsplittable* for $k \leq n/2$ and $A, B \in V$ if any two disjoint sets of k vertices containing A and B , respectively, have some edge between them. That is, G is (k, A, B) -unsplittable if for all partitions of the vertices of V into three disjoint sets V_1, V_2, V_3 of sizes $|V_1| = |V_2| = k$ and $|V_3| = n - 2k$ such that $A \in V_1$ and $B \in V_2$, there exists some edge $(u, v) \in E$ with $u \in V_1, v \in V_2$. From this definition we have immediately that G is (k, A, B) -unsplittable if and only if $G \not\subseteq_{A, B} \Lambda_k^{n-2k}$.

2.2 Secure Computation

Consider the scenario of n parties P_1, \dots, P_n with private inputs $x_1, \dots, x_n \in \mathcal{D}$ computing a function $f : \mathcal{D}^n \rightarrow \mathcal{D}^n$. Let Π be a protocol computing f . We consider security against adaptive t -threshold adversaries, that is, adversaries who corrupt a set of t parties, where $0 \leq t < n$.⁷ We assume the adversary to be semihonest or honest-but-curious. That is, the corrupted parties follow the prescribed protocol, but the adversary may try to infer additional information about the inputs of the honest parties. As noted in [36], in the computational setting, using zero-knowledge proofs, it is possible to generically compile a protocol which is secure against semihonest adversaries into another protocol which is secure against adaptive malicious adversaries [32].⁸ This justifies our focus on the semihonest setting here.

For a PPT adversary \mathcal{A} , let random variable $\text{REAL}_{\Pi, \mathcal{A}}^{x_1, \dots, x_n}$ consist of the views of the corrupted parties when the protocol Π is run on parties P_1, \dots, P_n with inputs x_1, \dots, x_n respectively. In the ideal world, the honest parties are replaced with a simulator \mathcal{S} which does not receive input values and knows only the output value of each corrupted party in an honest execution of the protocol. We define the random variable $\text{IDEAL}_{\Pi, \mathcal{A}, \mathcal{S}}^{x_1, \dots, x_n}$ as the output of the adversary \mathcal{A} in the ideal game with the simulator when the inputs to parties P_1, \dots, P_n are x_1, \dots, x_n respectively.

Definition 1. A protocol Π is said to t -securely compute the function f if

- For all $x_1, \dots, x_n \in \mathcal{D}^n$, party P_i receives y_i , where $(y_1, \dots, y_n) = f(x_1, \dots, x_n)$, at the end of the protocol.

⁷ Note that when $t = n$, there is nothing to prove.

⁸ We note that in the computational setting, it is also possible to transform, in a *black-box* way, a protocol which is secure against semihonest adversaries into another protocol which is secure against static malicious adversaries [35].

- For all adaptive semihonest PPT t -threshold adversaries \mathcal{A} , there exists a PPT simulator \mathcal{S} such that for all $x_1, \dots, x_n \in \mathcal{D}^n$

$$\left\{ \text{REAL}_{\Pi, \mathcal{A}}^{x_1, \dots, x_n} \right\} \equiv \left\{ \text{IDEAL}_{\Pi, \mathcal{A}, \mathcal{S}}^{x_1, \dots, x_n} \right\}$$

This definition is for secure computation with perfect information-theoretic security and a nonadaptive adversary. By [15], in the semihonest setting with information-theoretic security, any protocol which is nonadaptively secure is also adaptively secure. Consequently, satisfying this definition suffices to achieve adaptive security.

In the discussion below, we will sometimes relax security to statistical or computational definitions. A protocol is statistically t -secure if the random variables $\text{REAL}_{\Pi, \mathcal{A}}^{x_1, \dots, x_n}$ and $\text{IDEAL}_{\Pi, \mathcal{A}, \mathcal{S}}^{x_1, \dots, x_n}$ are statistically close, and computationally t -secure if they are computationally indistinguishable.

2.3 Oblivious Transfer

In this work, by OT, we refer to 1-out-of-2 oblivious transfer defined as follows.

Definition 2. We define 1-out-of-2 oblivious transfer f_{OT} for a sender $A = P_1$ with inputs $x_0, x_1 \in \{0, 1\}^m$, a receiver $B = P_2$ with input $b \in \{0, 1\}$ and $n - 2$ parties P_3, \dots, P_n with input \perp as

$$f_{\text{OT}}((x_0, x_1), b, \perp, \dots, \perp) = (\perp, x_b, \perp, \dots, \perp)$$

Note that while OT is typically defined as a 2-party functionality, the definition above adapts it our setting and formulates OT as an n -party functionality where only two parties supply non- \perp inputs.

Definition 3. We define a t -secure OT protocol, $\Pi_{A \rightarrow B}^{G, t}$, for an n -party network G with parties $A = P_1, B = P_2, P_3, \dots, P_n$ as one which t -securely computes the function f_{OT} on the inputs of the parties with A as the sender and B as the receiver.

We note that OT is symmetric, in the sense described below [63].

Lemma 1. [63] If there exists a t -secure OT protocol $\Pi_{A \rightarrow B}^{G, t}$ for an n -party network G with n parties $A = P_1, B = P_2, P_3, \dots, P_n$ with A as the sender and B as the receiver, then there exists a t -secure OT protocol $\widehat{\Pi}_{B \rightarrow A}^{G, t}$ for the same n parties with B as the sender and A as the receiver.

We represent parties as nodes of a graph G where an edge $\{A, B\}$ indicates that parties A and B may run a 1-secure OT protocol with A as the sender and B as the receiver. By Lemma 1, the roles of the sender and receiver may be reversed, so it makes sense to define G as an undirected graph.

We note the following result regarding the completeness of OT for achieving arbitrary secure multiparty computation.

Lemma 2. [46, 33, 45] Consider a network $G = (V, E)$ with $|V| = n$ such that $G \simeq K_n$. Then, for any function $f : \mathcal{D}^n \rightarrow \mathcal{R}^n$, there exists a protocol Π which $(n - 1)$ -securely computes the function f , where party i receives the i th input $x_i \in \mathcal{D}$ and produces the i th output $(f(x))_i \in \mathcal{R}$.

3 Warm-ups

Consider a network described by graph $G = (V, E)$ where the vertex set V has size n and represents n parties and each edge $\{P_i, P_j\} \in E$ indicates that parties P_i and P_j may run a 1-secure 2-party OT protocol with P_i as the sender and P_j as the receiver. Let $t < n$ be an upper bound on the number of corruptions made by the adversary. For which pairs of parties $A, B \in V$ does there exist a t -secure n -party OT protocol with A as the sender and B as the receiver?

For $t < n/2$ this is possible for all pairs of parties, since we can perform secure multiparty computation without any pre-existing OT channels if there is an honest majority [62]. Therefore, we focus on the setting where $t \geq n/2$. We initiate the study by looking at some known results.

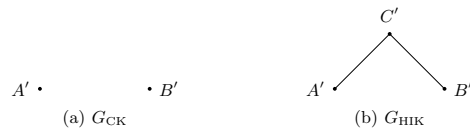


Fig. 1. Known impossibility results. Securely computing f_{OT} between A' and B' is impossible for $t = 1$ in G_{CK} and is impossible for $t = 2$ in G_{HIK} .

For $n = 2, t = 1$, we have from [17,49] that a 1-secure OT protocol (with perfect security) between the vertices of the two-vertex graph G does not exist unless the parties were already connected by an OT channel. This result is illustrated in Figure 1(a).

For $n = 3, t = 2$, we have from [36] that we can obtain a 2-secure OT protocol between a pair of vertices A, B only if those vertices are already connected by an OT channel, even if there are OT channels from both A and B to the third vertex C as depicted in Figure 1(b). More generally, [36] show that for any $n \geq 2$ and $t = n - 1$, there exists a t -secure OT protocol with sender A and receiver B only if those vertices are already connected by an OT channel, even if all other $\binom{n}{2} - 1$ pairs of vertices are connected by OT channels. This resolves the question for $n = 4, t = 3$. Thus, we begin our discussion with the case $n = 4, t = 2$.

In Figure 2 we present the key cases for the setting $n = 4, t = 2$. We will show below that these cases are sufficient to completely resolve the four-party setting. In the following discussion, vertex A will always be the sender in the OT protocol, and vertex B will always be the receiver.

3.1 Case 1 : Figure 2(a)

We show that for any network G such that $G \simeq_{A,B} G_1$ there does not exist a 2-secure OT protocol with A as the sender and B as the receiver.⁹ This is a

⁹ Recall that $H \simeq_{A,B} H'$ for two graphs H, H' if there exists an isomorphism between H and H' preserving the labels of vertices A and B .

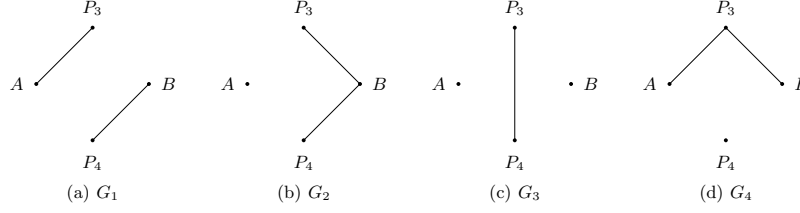


Fig. 2. Sufficient cases for 4 parties and 2 corruptions.

consequence of the impossibility result of [17,49]. The outline of the argument is as follows.

Consider components $\mathcal{C}_1 = \{A, P_3\}$ and $\mathcal{C}_2 = \{B, P_4\}$ of G , and let Π be a 2-secure protocol computing f_{OT} in G with A as the sender and B as the receiver. Then we can use Π to construct a 2-secure protocol Π' for the 2-party network in Figure 1(a) with A' as the sender and B' as the receiver. In protocol Π' , party A' runs Π for both parties of component \mathcal{C}_1 of G , and B' runs Π for both parties of component \mathcal{C}_2 . OT channel invocations can be handled locally, since all OT channels in G are between parties in the same component. Since protocol Π is 2-secure, in particular it is secure against corruptions of parties in \mathcal{C}_1 or the parties in \mathcal{C}_2 . Consequently Π' is a 1-secure OT protocol for a network $G' \simeq_{A',B'} G_{\text{CK}}$ with A' as the sender and B' as the receiver. However, from [17,49], we know that no such protocol exists with perfect security. Consequently there is no 2-secure protocol Π for a network $G \simeq_{A,B} G_1$.

Note that this impossibility holds not only for $G \simeq_{A,B} G_1$ but for any (A, B) -subgraph of G_1 . In particular, any four-vertex graph $G = (V, E)$ with $|E| \leq 1$ cannot have a 2-secure protocol computing f_{OT} between vertices A and B except in the trivial case when there is already an edge $\{A, B\} \in E$. This technique of reducing to the known impossibility results of [17,49,36] to obtain lower bounds is described formally in Section 4.

3.2 Case 2 : Figure 2(b)

In this example we obtain a positive result, showing that there exists a 2-secure OT protocol with A as the sender and B as the receiver. Let the *degree* of party P denote the degree of the corresponding vertex in the OT network. Since B has degree 2 in G_2 , we have that either B or at least one of its OT neighbors must be honest, and so one of the two OT channels must contain an honest party. This suggests the idea of using secret-sharing to ensure security against 2 corruptions.

Consider the following OT protocol where sender A has inputs $x_0, x_1 \in \{0, 1\}^m$ and receiver B has input $b \in \{0, 1\}$. A computes 2-out-of-2 shares (x_0^1, x_0^2) and (x_1^1, x_1^2) of its inputs x_0, x_1 , respectively. A then sends shares x_0^1 and x_1^1 to party P_3 and x_0^2 and x_1^2 to party P_4 . Parties P_3 and B invoke their secure OT channel with inputs (x_0^1, x_1^1) and b , and parties P_4 and B invoke their secure

OT channel with inputs (x_0^2, x_1^2) and b respectively. B uses the obtained shares x_b^1, x_b^2 to reconstruct x_b .

We informally argue the 2-security of this protocol assuming that exactly one of A and B is corrupt.¹⁰ Consider the case where A is corrupt and B is honest. The input of B is only used over secure OT channels, so by the 1-security of the OT channels with P_3 and P_4 , the corrupt parties can learn nothing about B 's input bit b . Now consider the case where B is corrupt and A is honest. Either P_3 or P_4 must be honest. If P_3 is honest then the security of OT channel $\{P_3, B\}$ implies that B learns nothing about share x_{1-b}^1 , so the security of the secret sharing scheme implies that the corrupt parties do not use x_{1-b} . By symmetry, the same argument applies if P_4 is honest. This completes the argument.

Note that by Lemma 1, we can also obtain a 2-secure OT protocol from A to B whenever A has degree 2 in OT network. Furthermore, we can extend this idea to construct a t -secure OT protocol whenever either the sender or the receiver has degree at least t . We call this protocol the t -claw protocol and describe it in detail in Section 5.1.

3.3 Case 3 : Figure 2(c)

Somewhat surprisingly, we can also show a positive result for graphs $G \simeq_{A,B} G_3$ even though the OT network has no edges involving either the sender A or the receiver B . The protocol is as follows. Since parties P_3 and P_4 have an OT channel between them, by Lemma 2, they can perform 1-secure MPC between them. They P_3 and P_4 use an MPC to compute 2-out-of-2 shares of OT correlations with uniformly random inputs and send corresponding shares to A and B who can then reconstruct the correlations. More concretely, the MPC protocol computes 2-out-of-2 shares $(r_0^1, r_0^2), (r_1^1, r_1^2)$ of two randomly sampled m -bit strings r_0, r_1 , 2-out-of-2 shares (c^1, c^2) of a random bit $c \in \{0, 1\}$, and independent 2-out-of-2 shares (s^1, s^2) of the string r_c . Party P_3 receives the first share of each secret, and party P_4 receives the second share. Party P_3 then sends shares r_0^1, r_1^1 to A and s^1, c^1 to B , while P_4 sends shares r_0^2, r_1^2 to A and s^2, c^2 to B . A can then reconstruct r_0 and r_1 , and B can reconstruct c and r_c . Parties A and B have now established a random OT correlation, which they can use to perform OT with their original inputs using OT correction [3].¹¹

We now informally argue the 2-security of this protocol. If A and B are both honest, then the corrupt parties receive no information about their inputs, while if A and B are both corrupt then there is nothing to prove. Consequently we can assume that exactly one of A and B is corrupt and that either P_3 or P_4 is

¹⁰ An additional step is needed to address the case in which A and B are both honest. Then P_3 and P_4 can both be corrupt and learn x_0 and x_1 , the inputs of A . This can be handled with the technique of OT correction, using a one-time pad and the secure point-to-point channel between A and B . Equivalently, we could run the protocol on random inputs, and then use method of [3] to obtain 1-out-of-2 OT from random OT. If A and B are both corrupt then there is nothing to prove.

¹¹ This OT correction step can be performed as follows. Party B sends $b' = b \oplus c$ to A . A responds with $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$. Finally, B computes $y_b \oplus r_c = x_b$.

honest. If A is corrupt and P_3 or P_4 is honest, then the adversary learns nothing about c and r_c , since it only sees one of the two shares of each. The OT correction phase uses these strings as one-time pads for inputs which are unknown to the adversary, and consequently are information-theoretically hidden from the adversary. Consequently A learns nothing about B . The case where B is corrupt and P_3 or P_4 is honest follows from the same argument.

This construction can be extended to obtain a t -secure OT protocol whenever the OT graph contains a t -clique consisting of t parties which are not the OT sender or receiver. We call this protocol the t -clique protocol and describe it in detail in Section 5.2.

3.4 Case 4 : Figure 2(d)

We also obtain a positive result for graphs $G \simeq_{A,B} G_4$. We introduce here a technique we call cascading. The idea is as follows. Using the protocol described in Section 3.2 for network G_2 of Figure 2(b), we have 2-secure OT protocol with P_3 as the sender and P_4 as the receiver. This effectively gives us an OT channel between P_3 and P_4 . Applying the protocol from Section 3.3 on augmented network, we now have a 2-secure OT protocol with A as the sender and B as the receiver. We describe this pictorially in Figure 3.

The 2-security of the protocol follows from the 2-security of the underlying protocols of Sections 3.2 and 3.3. The technique of cascading for combining t -secure protocols is described in detail in Section 5.3.

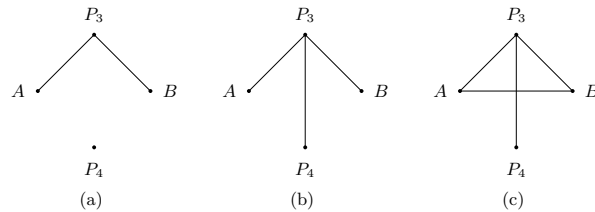


Fig. 3. Illustrating the cascading protocol for Case 4 : Figure 2(d); (a) \rightarrow (b) \rightarrow (c)

3.5 Cases 1–4 are exhaustive

Note that a t -secure OT protocol with sender A and receiver B in an OT network G trivially yields a t -secure protocol for any network G' such that $G \subseteq_{A,B} G'$. From cases 1 and 4, we can securely compute f_{OT} in a network G containing at most a single edge if and only if the edge is $\{A, B\}$ or $\{P_3, P_4\}$. From cases 1, 2, and 3, we can compute f_{OT} in a network G containing two or more edges including neither of $\{A, B\}$ or $\{P_3, P_4\}$ if and only if there is some vertex with degree at least 2 in the OT graph. This completes the characterization of 4-party networks with 2 corruptions.

4 Lower Bound

We now describe a family of impossibility results by a generic reduction to the impossibility result in [36], which we restate in our language below.

Lemma 3. [36] *Consider any three party network G with $G \simeq_{A',B'} G_{\text{HIK}}$, the graph in Figure 1(b). Then any 2-secure OT protocol with A' as the sender and B' as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A',B'} G_{\text{Kus}}$, the graph in Figure 1(a), with A' as the sender and B' as the receiver.*

The theorem below describes a family of impossibility results over a corresponding family of networks. We note that this result was observed in [36]; we restate it our language and provide a formal proof in Appendix B.

Theorem 1. *Let $n \geq 2$ and $n/2 \leq t < n$, and let G be an n party network such that $G \subseteq \Lambda_{n-t}^{2t-n}$, with $P_1 \in V_A$ and $P_2 \in V_B$. Any t -secure OT protocol for G with P_1 as the sender and P_2 as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A,B} G_{\text{Kus}}$ with A' as the sender and B' as the receiver.*

5 Building Blocks

In this section, we describe a few key protocols and techniques which we use in the subsequent sections to prove our main theorem.

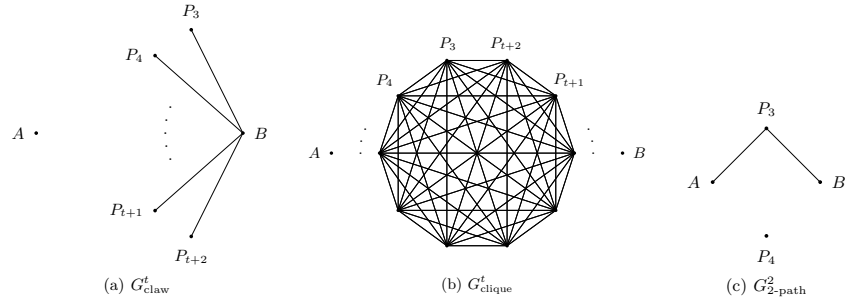


Fig. 4. Building block networks – (a) t -claw graph; (b) t -clique graph; (c) 2-path graph

5.1 The t -claw Protocol

The first protocol we describe is the t -claw protocol, where the graph G describing the network is such that $G \simeq_{A,B} G_{\text{claw}}^t$. The protocol is described in Protocol 1. The protocol is a straightforward generalization of the one described in Section

3.2. The idea is for A to compute t -out-of- t shares of its inputs and distribute them among the t parties connected to B . These t parties then perform OT with B so that B receives the shares to reconstruct his output.

Protocol 1: t -claw Protocol

Preliminaries: Let $A, B, P_3, \dots, P_{t+2}$ be the $t + 2$ parties in a network $G \simeq_{A,B} G_{\text{claw}}^t$. A has inputs $x_0, x_1 \in \{0, 1\}^m$ and B has input $b \in \{0, 1\}$.

Protocol:

1. B chooses a random bit $c \in \{0, 1\}$ and sends $b' = b \oplus c$ to A .
2. A chooses two random one-time pads $r_0, r_1 \in \{0, 1\}^m$ and sends $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ to B .
3. A then computes t -out-of- t shares (r_0^1, \dots, r_0^t) and (r_1^1, \dots, r_1^t) of r_0, r_1 , respectively.
4. For each $i \geq 3$, A sends shares r_0^i and r_1^i to party P_i .
5. For each $i \geq 3$, parties P_i and B execute the OT protocol $\Pi_{P_i \rightarrow B}^{G,1}$ with inputs (r_0^i, r_1^i) and c respectively.
6. B uses the obtained shares r_c^1, \dots, r_c^t to reconstruct r_c .
7. B finally computes $y_b \oplus r_c = x_b$.

Lemma 4. *Protocol 1 is an efficient t -secure OT protocol for a network $G \simeq_{A,B} G_{\text{claw}}^t$ with A as the sender and B as the receiver.*

Proof Intuition. The t -security of the protocol can be seen as follows. Steps 1, 2 and 7 perform OT correction, that is, they perform a random OT to 1-out-of-2 OT transformation. This transformation protects against the case that the parties P_3, \dots, P_{t+2} (that is, all but A and B) are corrupt. Suppose A were corrupt and B were honest. Clearly, A colluding with any of the parties P_3, \dots, P_{t+2} provides A with no additional information since all they possess are shares sent by A . Next, if A were honest and B corrupt, at least one of the parties P_3, \dots, P_{t+2} must be honest. B has no information about those shares and hence does not learn anything. Finally, if both A and B were corrupt, there is nothing to prove.

The full proof is deferred to Appendix C.

5.2 The t -clique Protocol

The next protocol we describe is the t -clique protocol, where the graph G describing the network is such that $G \simeq_{A,B} G_{\text{clique}}^t$. The protocol is described in Protocol 2. The protocol is a straightforward generalization of the one described in Section 3.3. The idea is for the parties P_3, \dots, P_{t+2} to compute t -out-of- t shares of OT correlations and send them to A and B respectively. This is done via multiparty computation since the parties have a complete network of OT

channels (Lemma 2). A and B then perform OT correction using their secure channel.

Protocol 2: t -clique Protocol

Preliminaries: Let $A, B, P_3, \dots, P_{t+2}$ be the $t+2$ parties in a network $G \simeq_{A,B} G_{\text{clique}}^t$. A has inputs $x_0, x_1 \in \{0, 1\}^m$ and B has input $b \in \{0, 1\}$.

Protocol:

1. Parties P_3, \dots, P_{t+2} use their pairwise OT channels to run t -secure MPC for the function f using the protocol from Lemma 2 for the function f described ahead. The function f is to securely compute t -out-of- t shares (r_0^1, \dots, r_0^t) , (r_1^1, \dots, r_1^t) of two randomly sampled one-time pad keys r_0, r_1 , (c^1, \dots, c^t) of a random bit $c \in \{0, 1\}$, and independent shares (s^1, \dots, s^t) of key r_c , so that party $i+2$ receives only shares r_0^i, r_1^i, s^i, c^i for each i .
2. Each party P_{i+2} for $i \geq 1$ sends shares r_0^i, r_1^i to A and s^i, c^i to B .
3. A uses shares (r_0^1, \dots, r_0^t) and (r_1^1, \dots, r_1^t) to reconstruct r_0 and r_1 .
4. B uses shares (c^1, \dots, c^t) and (s^1, \dots, s^t) to reconstruct c and r_c and sends $b' = b \oplus c$ to A .
5. A computes $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ and sends both to B .
6. B computes $y_b \oplus r_c = x_b$.

Lemma 5. *Protocol 2 is an efficient t -secure OT protocol for a network $G \simeq_{A,B} G_{\text{clique}}^t$ with A as the sender and B as the receiver.*

Proof Intuition. The t -security of the protocol can be seen as follows. Steps 4, 5 and 6 perform OT correction, that is, they perform a random OT to 1-out-of-2 OT transformation. This transformation protects against the case that the parties P_3, \dots, P_{t+2} (that is, all but A and B) are corrupt. If one of A and B were corrupt, there exists at least one honest party among the parties P_3, \dots, P_{t+2} . Hence, even by colluding, A or B would have no information about those shares and would not learn anything. Finally, if both A and B were corrupt, there is nothing to prove.

The full proof is deferred to Appendix C.

5.3 Cascading

The following building block is a generalization of the technique described in Section 3.4. The technique describes a general method of combining protocols iteratively. In our context, this can be thought of a tool for transforming a network described by a graph G to one described by a graph G' , where $G \subseteq_V G'$ and G and G' are both graphs on the same vertex set V . In other words, it describes protocols as adding new edges indicating the establishment of OT

correlations between new pairs of parties in the network. With this abstraction, it is easy to view the technique of cascading as one which combines protocols iteratively to transform the underlying network by adding new edges. This is described formally below.

Definition 4. Let $G = (V, E)$ and $G' = (V, E')$ be two graphs on the same set of vertices, V , with $G \subseteq_V G'$. We say that a protocol Π t -transforms a network G into the network G' if for each $\{P_i, P_j\} \in E' \setminus E$, Π is a t -secure OT protocol for a network G with P_i as the sender and P_j as the receiver.¹²

Lemma 6. If Π_1 is a protocol that runs in time T_1 and t -transforms network G_1 into G_2 , and Π_2 is a protocol that runs in time T_2 and t -transforms network G_2 into G_3 , then there exists a protocol Π that runs in time $T_1 T_2$ and t -transforms G_1 into G_3 .

Proof. The protocol Π simply runs Π_2 , running protocol Π_1 to obtain correlations whenever Π_2 invokes OT on an edge of $G_2 \setminus G_1$. Let \mathcal{S}_1 and \mathcal{S}_2 be the simulators associated with Π_1 and Π_2 respectively. The simulator for Π simply runs \mathcal{S}_2 , invoking \mathcal{S}_1 for OT calls made on edges in $G_2 \setminus G_1$. \square

Using OT extension [2,43], we can also obtain a computationally secure version of cascading with improved efficiency.

Lemma 7. Let λ be a computational security parameter. Assuming one-way functions or correlation-robust hash functions, if Π_1 is a protocol that runs in time T_1 and t -transforms network G_1 into G_2 , and Π_2 is a protocol that runs in time T_2 and t -transforms network G_2 into G_3 , then there exists a computationally secure protocol Π that runs in time $\lambda \cdot T_1 + T_2 \cdot \text{poly}(\lambda)$ and t -transforms G_1 into G_3 .

Proof. First, run protocol Π_1 λ times on random inputs to obtain λ independent OT correlations for each edge of $G_2 \setminus G_1$. Then run Protocol Π_2 , using OT extension [2,43] to obtain OT correlations for OT calls made on edges in $G_2 \setminus G_1$. \square

5.4 The 2-path graph

The protocol described in this section is a commonly used subroutine in several of the protocols which follow. It is a particular combination of the tools encountered in Sections 5.1, 5.2 and 5.3). The subroutine, which we call 2-path, is the same as the one described in Section 3.4. It is used to obtain OT correlations between parties who have a common neighbor in a four-party network with at most two corruptions (see Figure 4(c)).

Lemma 8. Protocol 3 is an efficient 2-secure OT protocol for a network $G \simeq_{A,B} G_{2\text{-path}}^2$ with A as the sender and B as the receiver.

¹² Note that a single protocol Π may set up independent OT correlations for several pairs of parties $\{P_i, P_j\} \in E' \setminus E$.

Proof. This follows immediately from Lemma 6 and the 2-security of Protocols 1 and 2 for $t = 2$ (Lemmata 4 and 5). \square

Protocol 3: 2-path

Preliminaries: Let A, B, C, D be the parties, and let there exist OT channels (A, C) and (B, C) . A has input (x_0, x_1) , and B has input $b \in \{0, 1\}$.

Protocol:

1. Invoke Protocol 1 (2-claw) on parties (D, C, A, B) to obtain OT correlations on edge (D, C) .
2. By Lemma 6, we have an OT channel between D and C .
3. Invoke Protocol 2 (2-clique) on parties (A, B, C, D) .

5.5 Combiners

The notion of OT combiners is one which aims at combining several candidate protocols for establishing OT correlations between two parties with the property that a majority of them remain secure in the presence of any adversary \mathcal{A} from a class of adversaries \mathbb{A} into a single protocol which remains secure in the presence of any adversary from the same class \mathbb{A} . The following lemma is due to [56,37], relying on prior work by [38,64] based on a construction by [22].

Lemma 9. [56,37] *Let \mathbb{A} be an adversary class. Suppose there exist m protocols Π_1, \dots, Π_m for $f_{OT}(A, B, P_1, \dots, P_n)$ such that for any adversary $\mathcal{A} \in \mathbb{A}$ a majority of the protocols are secure. Then, there exists a protocol $\Pi^*(\Pi_1, \dots, \Pi_m)$ for $f_{OT}(A, B, P_1, \dots, P_n)$ which is secure against all adversaries $\mathcal{A} \in \mathbb{A}$. Moreover, if each protocol Π_i is efficient and perfectly secure, then so is Π^* .*

6 The case $t = n/2$

We now consider the specific case of $t = n/2$, that is, at most half the parties are corrupt. We note that this is the smallest value of t for which the question is non-trivial. From the lower bounds proven in Theorem 1, we already have that for all n -party networks G containing A and B such that $G \subseteq_{A,B} \mathcal{A}_{n/2}^0$, there exists no $n/2$ -secure OT protocol with A as the sender and B as the receiver. Quite surprisingly, we show ahead, in Theorem 2, that those networks are the only ones where there exists no such protocol. In fact, we show an explicit $n/2$ -secure OT protocol with A as the sender and B as the receiver whenever the network G is $(n/2, A, B)$ -unsplittable.

Theorem 2. Consider an n -party network G , which contains A and B as two of the parties. Protocol 5 is an $n/2$ -secure OT protocol with A as the sender and B as the receiver if and only if G is $(n/2, A, B)$ -unsplittable.

We analyze the efficiency of the protocol in Theorem 6 in Appendix D. The protocol as stated runs in quasi-polynomial time. We can also obtain a computationally secure protocol which runs in polynomial time. The protocol we describe proceeds in two stages. In the first stage, the protocol transforms every connected component of the network into a clique. This transformation is very specific to the case of $t = n/2$, in particular, that each connected component can actually function as a clique is true only in this case. This transformation is carried out by means of repeatedly calling a protocol we call “Completing Triangles” which obtains OT correlations between parties who have a common neighbour. We have already seen a special case of this protocol in the analysis for $n = 4$ and $t = 2$ in Section 5.4. In fact, the protocol which achieves the same goal in the case of general $n \geq 4$ and $t = n/2$ uses the building block Protocol 3 along with the building block of OT combiners described in Section 5.5. Protocol 4 describes the protocol in order to achieve this.

Protocol 4: Completing Triangles

Preliminaries: Let A, B, C, P_4, \dots, P_n be the n parties, and let there exist OT channels (A, C) and (B, C) . A has input (x_0, x_1) , and B has input $b \in \{0, 1\}$.

Protocol:

1. Run a combined protocol $\Pi^*(\Pi_4, \dots, \Pi_n)$ on the $n - 3$ protocols Π_4, \dots, Π_n , where
 - For each $i \geq 4$, Π_i denotes an invocation of Protocol 3 (2-path) with the four parties A, B, C, P_i with A as the sender and B as the receiver.

Lemma 10. Let G be an n -vertex OT network with edges $\{A, C\}$ and $\{B, C\}$. Protocol 4 is an $n/2$ -secure OT protocol for the network G with A as the sender and B as the receiver.

Proof. We consider cases depending on the number of corrupted parties in the set $T = \{A, B, C\}$. If T contains at most one corrupted party, then each tuple (A, B, C, P_i) for $i \geq 4$ contains at most 2 corrupted parties, so each protocol Π_i in step 1 is secure. If T contains two corrupted parties, then there are at most $t - 2 = (n - 4)/2$ corrupted parties among P_4, \dots, P_n , so a majority of these parties are honest. Consequently a majority of the protocols Π_i which are combined in step 1 are secure. Thus, in either case, by Lemma 9 the protocol is secure. Finally, if all three parties of T are corrupted, then all uncorrupted parties receive no input, so the simulator \mathcal{S} can perfectly simulate the uncorrupted parties by running the honest protocol. Therefore Protocol 4 is $n/2$ -secure. \square

Proof Intuition (Theorem 2): It is easy to see that by executing Protocol 4 repeatedly, one can obtain OT correlations between any pair of parties in the same connected component. In other words, for $t = n/2$, we can assume that we are given a network which consists of disjoint cliques (Lemma 6). This is done in step 1 of Protocol 5. Hence, if A and B were in the same connected component in G , this process would end up with correlations between A and B and we can terminate the protocol (step 2).

Assume this is not the case. A natural next step to try is to run the clique protocol described in Section 5.2 with each of the cliques and parties A and B with the intent of setting up OT correlations between A and B . The troubling aspect, however, is that we are unable to fix the parameter t in any of the cliques. Indeed, in many of these invocations, the number of corrupted parties may be more than what Protocol 2 can handle in order to guarantee security. However, for an invocation to be secure, we only require that the clique contains at least one honest party. This is because we can assume without loss of generality that at least one of A or B is honest since otherwise we have nothing to prove. But now, this gives us that a majority of the cliques actually contain at least one honest party. Hence, if we invoke Protocol 2 for each of the parties on their respective cliques, a majority of them would be secure and now we can combine these candidate invocations to obtain a secure protocol following Lemma 9. This is performed in step 5 of the Protocol 5. Finally, we note that steps 3, 4 and 6 perform OT correction, that is, they perform a random OT to 1-out-of-2 OT transformation. This describes $n/2$ -security of Protocol 5.

Proof (Theorem 2). The “only if” part of theorem has been proved by virtue of the lower bound proven in Theorem 1 with $t = n/2$. We now prove the “if” part. We note that in the case where A and B are in the same connected component in the network G , by the $n/2$ -security of Protocol 4 and Lemma 6, we note that Protocol 5 is an $n/2$ -secure OT protocol with A as the sender and B as the receiver, thus proving the theorem.

We now proceed to the case where A and B are not in the same connected component in G . We must show that the protocol is secure against t -threshold adversaries as long as the vertices cannot be partitioned into two sets V_A, V_B each of size $t = n/2$ with $A \in V_A, B \in V_B$ such that there are no edges between V_A and V_B . Let \mathcal{A} be a t -threshold adversary which corrupts parties T , $|T| \leq t$. We will construct a simulator \mathcal{S} which plays the role of the uncorrupted parties.

If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so the simulator can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \emptyset$ then \mathcal{S} chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend on the input at all are on point-to-point channels between A and B , the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that the corrupted parties T include exactly one of A, B . If $A \in T$ but $B \notin T$, then \mathcal{S} chooses an arbitrary bit b and runs the protocol, invoking the OT simulator for each invocation of Protocol 4. It follows that as long as the combined protocol Π^* in step 5 is secure against \mathcal{A} , Protocol 5 is secure against \mathcal{A} . It remains to show that a majority of the n protocols

Protocol 5: $n/2$ corruptions

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \dots, P_n$ be the n parties in a network $G = (V, E)$. A has input (x_0, x_1) , and B has input $b \in \{0, 1\}$.

Protocol:

1. While there exist parties $P_i, P_j, P_k \in V$ such that $\{P_i, P_j\} \in E, \{P_j, P_k\} \in E$, but $\{P_i, P_k\} \notin E$:
 - (a) Let S be the set of triples of distinct vertices $(X, Y, Z) \in V^3$ with $\{X, Y\} \in E, \{Y, Z\} \in E$, and $\{X, Z\} \notin E$.
 - (b) For each triple $(X, Y, Z) \in S$, invoke Protocol 4 with independent random inputs $(r_0^{i,k}, r_1^{i,k})$ and $b^{i,k}$, to obtain OT correlations along edge $\{X, Z\}$.
 - (c) Invoking cascading (Lemma 6), we can add $\{X, Z\}$ to the edge set E for all triples $(X, Y, Z) \in S$.

The OT network G now consists of disjoint cliques $\mathcal{C}_1, \dots, \mathcal{C}_\ell$.
2. If A and B are in the same clique, then halt.
3. B samples a random bit c and sends $b' = b \oplus c$ to A .
4. A chooses random one-time pads r_0, r_1 and sends $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ to B .
5. Let \mathcal{C}_1 be the clique containing A and \mathcal{C}_2 be the clique containing B . For each party $P_i, i \geq 3$, let $\mathcal{C}(i)$ denote the clique containing party i , and let $P_{j_1}, \dots, P_{j_{|\mathcal{C}(i)|}}$ denote the parties in clique $\mathcal{C}(i)$. Run a combined protocol $\Pi^*(\Pi_1, \dots, \Pi_n)$ on the n protocols Π_1, \dots, Π_n , where
 - For each $i \in [n]$, Π_i denotes an invocation of Protocol 2 on the $|\mathcal{C}(i)| + 2$ parties $A, B, P_{j_1}, \dots, P_{j_{|\mathcal{C}(i)|}}$ with inputs (r_0, r_1) and c .^a
6. Finally, B computes $x_b = y_b \oplus r_c$.

^a In the case $\mathcal{C}(i) = \mathcal{C}_1$, A is both the OT sender and a member of the clique. A similar condition holds for B in the case $\mathcal{C}(i) = \mathcal{C}_2$.

Π_1, \dots, Π_n are secure against \mathcal{A} . Since party B is honest, by Lemma 5, protocol Π_i is secure against \mathcal{A} as long as at least one of the parties in clique $\mathcal{C}(i)$ is honest. In particular, if party P_i is honest then protocol Π_i is secure against \mathcal{A} . At most t of the parties P_1, \dots, P_n are corrupt, so the only protocols which may be insecure against \mathcal{A} are the t protocols Π_i corresponding to the corrupted parties P_i . Assume that all t of these protocols are insecure against \mathcal{A} . We then have the corrupted parties lie in completely corrupted cliques who sizes sum up to $n/2$. This then gives a set $V_A = T$ of $n/2$ parties containing A but not B such that there are no edges from V_A to the remaining vertices $V_B = \bar{T}$. However, we know that G possesses no such partition. Hence, at most $t-1 < n/2$ of the n protocols are insecure against \mathcal{A} and hence by Lemma 9, the combined protocol Π^* in step 5 is secure and hence Protocol 5 is secure against \mathcal{A} .

The remaining case that $B \in T$ but $A \notin T$ is similar. Here, the simulator \mathcal{S} is given the output value x_b . \mathcal{S} runs the protocol with (x_b, x_b) as the input to A , again invoking the OT simulator for each invocation of Protocol 4. As above,

as long as the combined protocol Π^* in step 5 is secure against \mathcal{A} , Protocol 5 is secure against \mathcal{A} . By the same argument, the only protocols Π_i which may be insecure against \mathcal{A} are the t protocols corresponding to the corrupted parties P_i . If all t of these protocols are insecure against \mathcal{A} , as above, we have a set $V_A = \overline{T}$ of $n/2$ parties containing A but not B such that there are no edges from V_A to the remaining vertices $V_B = T$. However, we know that G possesses no such partition. Hence, at most $t - 1 < n/2$ of the n protocols are insecure against \mathcal{A} and hence by Lemma 9, the combined protocol Π^* in step 5 is secure and hence Protocol 5 is secure against \mathcal{A} . \square

7 The case $t = n - 2$

On account of the lower bound proven in [36], we note that $t = n - 2$ is the largest value of t which the question is non-trivial. In this section we present an improved computationally efficient protocol for the special case $t = n - 2$ for all networks G with A as the sender and B as the receiver whenever the network G satisfies G is $(2, A, B)$ -unsplittable.

Theorem 3. *Consider an n -party network G , which contains A and B as two of the parties. Let $t = n - 2$. Protocol 6 is an efficient t -secure OT protocol with A as the sender and B as the receiver if and only if G is $(2, A, B)$ -unsplittable.*

Protocol 6: $n - 2$ corruptions

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \dots, P_n$ be the n parties, and let graph $G = (V, E)$ be the OT network among the parties. A has input (x_0, x_1) , and B has input $b \in \{0, 1\}$.

Protocol:

1. For all pairs of parties $P_i, P_j \in V$ with $i, j \geq 3$ such that $\{P_i, P_j\} \notin E$:
 - (a) Invoke Protocol 5 (or any 2-secure protocol for $n' = 4$) on the induced OT subgraph $G_{i,j} := G \cap \{P_i, P_j, A, B\}$ with independent random inputs $(r_0^{i,j}, r_1^{i,j})$ and $b^{i,j}$, to obtain OT correlations along edge $\{P_i, P_j\}$.
 - (b) By virtue of cascading (Lemma 6), we can add edge $\{P_i, P_j\}$ to the graph G .^a

The OT network G now contains a $(n - 2)$ -clique among vertices P_3, \dots, P_n .
2. Invoke Protocol 2 (t -clique) with input (x_0, x_1) and b .

^a We will only have OT security over this edge when at least two of the parties P_i, P_j, A, B are honest, but we obtain the functionality of the edge regardless. We address security of the overall protocol in the proof.

The protocol is built upon the following structural aspect of the network G under consideration. We are only concerned with networks G that are

$(2, A, B)$ -unsplittable. This means that for any two sets vertices V_A and V_B such that $|V_A| = |V_B| = 2$, $A \in V_A$ and $B \in V_B$, there exists an edge crossing the sets V_A and V_B . In particular, by definition, this implies that for any two parties P_i, P_j where $i, j \geq 3$, the sub-network $G_{i,j}$ induced by the parties A, B, P_i and P_j is $(2, A, B)$ -unsplittable. Flipping the argument on its head, this also means that for any two parties P_i, P_j , $G_{i,j}$ is $(2, P_i, P_j)$ -unsplittable. Hence, we could try to obtain OT correlations between every pair of vertices P_i, P_j by running Protocol 5 on every $G_{i,j}$ for $n = 4$ parties. Notice that if these invocations were secure, then we would obtain an $(n - 2)$ -clique in the network after which we can execute Protocol 2 in order to obtain OT correlations between A and B . This describes Protocol 6. The only concern however is that each of the executions of Protocol 5 would be secure only if each of them contained at most two corrupt parties. While this need not be true in general, and hence we cannot leverage the security of the executions of Protocol 5, we will argue that Protocol 6 still remains secure against $t = n - 2$ corruptions.

Proof Intuition (Theorem 3): We take off from the description of Protocol 6 above. It is easy to see that in order to analyze the $(n - 2)$ -security of Protocol 6, we need to analyze whether the invocations of Protocol 5 on sub-networks $G_{i,j}$ are secure. In particular, we know if at most two of the four parties in $G_{i,j}$ are honest, then that particular invocation of Protocol 5 is secure and yields secure OT correlations between the parties P_i and P_j . And then, appealing to Lemma 6, we have that the network G now possesses the edge $\{P_i, P_j\}$.

Note however that each $G_{i,j}$ has at least one honest party since at most one of A or B is corrupt (otherwise, there is nothing to prove). We now consider a sub-network $G_{i,j}$ in which three of the parties are corrupt. Since at least one of A or B is honest, this would mean that both P_i and P_j are corrupt. Thus, there is nothing to prove regarding the security of the invocation of Protocol 5 on $G_{i,j}$ since we are looking to establish OT correlations between P_i and P_j and they are both corrupt. Combining these claims, we have that each of the invocations of Protocol 5 is secure and yields secure OT correlations between the pairs of parties P_i, P_j for all $i, j \geq 3$. By virtue of Lemma 6, we obtain an $(n - 2)$ -clique in the network and the $(n - 2)$ -security of Protocol 2 with $t = n - 2$ proves the $(n - 2)$ -security of Protocol 6.

The full proof of the theorem is deferred to Appendix E.

8 The General Case: $t \geq n/2$

In this section, we investigate the question for general $t \geq n/2$. Note that from the protocols in Sections 6 and 7 we already have tight answers for the cases $t = n/2$ and $t = n - 2$. We address the question from both ends of the spectrum, namely for t larger than $n/2$ and t smaller than $n - 2$. These analyses culminate in the descriptions of two distinct protocols using the protocols from Sections 6 and 7 as their respective base cases. In particular, we note that the protocols we describe are efficient closer to their respective ends of the spectrum. That

is, the protocol described in Section 8.1 is quasi-polynomially efficient¹³ when $t = n/2 + \mathcal{O}(1)$, while the protocol described in Section 8.2 is efficient when $t = n - \mathcal{O}(1)$. We describe these protocols ahead and observe that the combination of these protocols yields one which is efficient under computational security when either $t = n/2 + \mathcal{O}(1)$ or $t = n - \mathcal{O}(1)$. We note that the problem of recognizing whether there exists a t -secure OT protocol is efficient in these cases, while the recognition problem for general n, t is coNP-complete.

8.1 General Protocol (Quasi-polynomial for $t = n/2 + \mathcal{O}(1)$)

We now describe a t -secure OT protocol for all networks G with A as the sender and B as the receiver whenever the network G is $(n-t, A, B)$ -unsplittable. Notice that on account of the lower bound described in Section 4, this result is tight.

Theorem 4. *Consider an n -party network G which contains parties A and B . Let $t \geq n/2$. Protocol 7 is a t -secure OT protocol with A as the sender and B as the receiver if and only if G is $(n-t, A, B)$ -unsplittable. The protocol achieves perfect security and runs in quasi-polynomial time for $t = n/2 + \mathcal{O}(1)$. Assuming one-way functions, we can also obtain a protocol which achieves computational security and runs in polynomial time for $t = n/2 + \mathcal{O}(1)$.*

The main idea behind the protocol is recursion, that is, to reduce the problem of obtaining an OT protocol on an n -vertex graph with $t > n/2$ corrupted parties to a number of instances of $(n-1)$ -vertex graphs, most of which have at most $t-1$ corrupted parties. An important point to note, which we prove ahead, is that the $(n-1)$ -vertex sub-graphs, say G' , have structure similar to G , in the sense that, G' is $(n'-t', A, B)$ -unsplittable if G is $(n-t, A, B)$ -unsplittable, where $n' = n-1$ and $t' = t-1$. We can now try the natural strategy of recursing on these smaller problem instances and invoking an OT combiner to obtain the final protocol.

More precisely, the protocol constructs $n-2$ instances of subgraphs on $n-1$ vertices each where each one is obtained by deleting exactly one of the vertices other than A and B . It is these sub-graphs which preserve the structure in G , as described above. The candidate $n-2$ protocols, each run on one of the subgraphs, are $(t-1)$ -secure OT protocols with A as the sender and B as the receiver. The final protocol is to simply run the protocol which combines all the candidate protocols. What remains to be proven is that a majority of the subgraphs defined actually possesses at most $t-1$ corrupt parties.

Proof Intuition (Theorem 4): We may assume that at least one of A or B is honest since otherwise there is nothing to prove. As described above, we wish to argue that a majority of the sub-graphs defined actually possesses at most $t-1$ corrupt parties. Note that this claim combined with the claim that these specially chosen sub-graphs preserve the structure of G (that is, for any of these

¹³ or polynomially efficient under computational security

sub-graphs G' , G' is $(n - t, A, B)$ -unsplittable if G is $(n - t, A, B)$ -unsplittable) complete the proof.

However, this claim follows from the following trivial observation. Since $t > n/2$, if exactly t parties are corrupt then a majority of the sub-graphs definitely contain at most $t - 1$ corrupt parties since A and B are not both corrupt. If strictly fewer than t parties are corrupt then all of the sub-graphs contain at most $t - 1$ corrupt parties. In either case, for a majority of sub-graphs, at most $t - 1$ of the parties are corrupt. Invoking Lemma 9 completes the argument.

Protocol 7: General Protocol I

Preliminaries: Let A, B, P_3, \dots, P_n be the n parties in a network G and let $t \geq n/2$ be the maximum number of corruptions. A has input (x_0, x_1) , and B has input $b \in \{0, 1\}$.

Protocol:

1. If $t = n/2$, then invoke Protocol 5 and halt.
2. Otherwise, run a combined protocol $\Pi^*(\Pi_3, \dots, \Pi_n)$, where
 - For each $i \geq 3$, Π_i denotes the recursive invocation of this protocol on the $n - 1$ parties excluding party P_i with the induced sub-network $G \setminus \{P_i\}$ and $t' = t - 1$ corruptions.

We first present a structure lemma which is proven in Appendix E.

Lemma 11. *Given graph $G = (V, E)$ and a vertex i , let G_i be the induced graph on the $n - 1$ vertices $V \setminus \{i\}$. If G is $(n - t, A, B)$ -unsplittable, then G_i is also $(n - t, A, B)$ -unsplittable.*

Using the lemma, we now prove Theorem 4.

Proof (Theorem 4:). The “only if” part of theorem has been proved by virtue of the lower bound proven in Theorem 1. The efficiency claim follows immediately from Theorem 6. We now prove the “if” direction by induction on $2t - n$. The base case of $2t - n = 0$ follows from Theorem 2. Now assume for induction that the statement holds for $n' = n - 1, t' = t - 1$ with $t > n/2$.

Let \mathcal{A} be a t -threshold adversary which corrupts parties T , $|T| \leq t$. If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so the simulator can perfectly simulate the uncorrupted parties. Consequently it suffices to consider the case $\{A, B\} \cap T \leq 1$. By Lemma 11, each sub-network $G \setminus \{P_i\}$ is $(n' - t', A, B)$ -unsplittable. For each $i \geq 3$, let $T_i = T \setminus \{P_i\}$ denote the corrupted parties in sub-network $G \setminus \{P_i\}$. If $|T| < t$ then $|T_i| \leq |T| \leq t'$ for all i . Otherwise, $|T| = t$. In this case, since $\{A, B\} \cap T \leq 1$, it follows that at least $t - 1$ of the $n - 2$ parties P_3, \dots, P_n are corrupted. Since $t > n/2$, this is a majority of the parties P_3, \dots, P_n . For each corrupted party P_i , $|T_i| = t'$.

Hence, in each case, we have that a majority of the sets T_3, \dots, T_n satisfy $|T_i| \leq t'$. Therefore by our inductive assumption, a majority of the protocols Π_3, \dots, Π_n in step 2 are secure against \mathcal{A} . Consequently, by Lemma 9, we have that the combined protocol Π^* is secure against \mathcal{A} . Hence, by induction, the theorem holds for all n, t with $t \geq n/2$. \square

We have the immediate consequence that in order to obtain a complete network and perform secure multiparty computation, for every pair of parties A, B , the condition described in Theorem 4 is necessary and sufficient.

Corollary 1. *Let G be an n -party network. For $t \geq n/2$, we can t -securely generate OT correlations between all pairs of parties (thus, completing the OT network) if and only if the G is $(n - t)$ -unsplittable.*

A proof of the corollary is given in Appendix E.

8.2 General Protocol (Efficient for $t = n - \mathcal{O}(1)$)

We now describe another t -secure OT protocol for all networks G with A as the sender and B as the receiver whenever the network G is $(n - t, A, B)$ -unsplittable. This protocol uses, in spirit, a reduction in the opposite sense than the one described in Section 8.1. The protocol is efficient whenever $t = n - \mathcal{O}(1)$.

Protocol 8: General protocol II

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \dots, P_n$ be the n parties in a network $G = (V, E)$. A has input (x_0, x_1) , and B has input $b \in \{0, 1\}$. Let $k = n - t$.

Protocol:

1. Invoke Protocol 6 with $t' = n - 2$ on the n' -node network G' with inputs (x_0, x_1) and b , where $n' = \binom{n-2}{k-1} + 2$, and
 - S_{k-1} is the set of subsets of $\{P_3, \dots, P_n\}$ of size $k - 1$.
 - The n' vertices of G' correspond to A, B , and the $\binom{n-2}{k-1}$ subsets of S_{k-1} .
 - The edges of G' are defined as follows. Two subsets $X, Y \in S_{k-1}$ will have an edge if either $X \cap Y \neq \emptyset$ or there exists a pair of parties $P_i \in X$ and $P_j \in Y$ with $\{P_i, P_j\} \in E$.
 - Invocation fo OT over an edge $\{X, Y\}$ in G' with inputs (z_0, z_1) and c is performed as follows.
 - If $X \cap Y \neq \emptyset$, then choose some party $P_i \in X \cap Y$. $P_i \in X$ and hence knows (z_0, z_1) ; similarly, $P_i \in Y$ and knows c . Consequently P_i knows z_c , and sends it to the other members of set Y .
 - If $X \cap Y = \emptyset$, there is a pair of parties $P_i \in X, P_j \in Y$ such that $\{P_i, P_j\} \in E$. P_i knows (z_0, z_1) and P_j knows c , so they can invoke OT over the channel (P_i, P_j) in G , and P_j can then send the value z_c to the other members of set Y .

Theorem 5. *Consider an n -party network G , which contains A and B as two of the parties. Let $t \geq n/2$. Protocol 8 is a t -secure OT protocol with A as the sender and B as the receiver if and only if G is $(n - t, A, B)$ -unsplittable. The protocol is efficient for $t = n - \mathcal{O}(1)$.*

The idea behind this protocol is the following. We blow up the network in order to obtain a large number, N , of parties apart from A and B such that at least one of them is guaranteed to be honest. With this guarantee in mind, the protocol description is straightforward. We may assume that at least one of A and B is honest, as otherwise there is nothing to prove. Now, the total number of parties is $n' = N + 2$ and the number of honest parties is at least 2 since one of A or B and one of the other parties is guaranteed to be honest. This corresponds to the case that $t' = n' - 2$. We can now apply the protocol from Section 7. What remains to be discussed is the construction of these parties, a structural lemma that if the network G we began with were $(n - t, A, B)$ -unsplittable, then the newly constructed network G' is $(2, A, B)$ -unsplittable, and why G' is guaranteed to have at least two honest parties, in particular, why one of the parties other than A and B is honest.

Proof Intuition (Theorem 5): We first describe the new network generated by Protocol 8. The parties other than A and B in the newly constructed network are constructed as subsets of the parties in G other than A and B . The sizes of these subsets are $n - t - 1$, that is, in G' , the parties other than A and B are all $(n - t - 1)$ -size subsets of the remaining parties in G . It can be proved that this new network G' is $(2, A, B)$ -unsplittable if G is $(n - t, A, B)$ -unsplittable, where the edges of G' are as in described in Protocol 8 (Lemma 12). A party X , which is formed as a $(n - t - 1)$ -size subset will be considered honest if all constituent parties $P_i \in X$ are honest. Now, the reason for there being one honest party among these $(n - t - 1)$ -size subset parties is the following trivial observation. Since one of A and B is honest and at most t parties are corrupt, at least $n - t$ parties are honest and in particular, at least $n - t - 1$ of the non- A, B parties must be honest. This means that one of the subsets is completely honest. In particular, one of the parties other than A and B are honest in G' and hence G' is guaranteed to have at least two honest parties. Combining these facts and invoking Theorem 3 completes the argument.

We will use the following structural lemma about the network G' constructed in Protocol 8.

Lemma 12. *If G is $(n - t, A, B)$ -unsplittable, then G' is a $(2, A, B)$ -unsplittable network on $n' = \binom{n-2}{n-t-1} + 2$ vertices, where G' is the network produced in Protocol 8.*

The full proofs of Lemma 12 and Theorem 5 are deferred to Appendix E.

References

1. B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In *Crypto (2)*, pages 166–184, 2013.
2. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *STOC*, pages 479–488, 1996.
3. Donald Beaver. Precomputing oblivious transfer. In *Crypto*, pages 97–109, 1995.
4. Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. In *TCC*, pages 213–230, 2008.
5. A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *Crypto*, pages 80–97, 1999.
6. A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In *Crypto*, pages 538–557, 2010.
7. M. Bellare, V.T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE Security and Privacy*, pages 478–492, 2013.
8. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computations. In *STOC*, pages 1–10, 1988.
9. E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Crypto*, pages 663–680, 2012.
10. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *Eurocrypt*, pages 169–188, 2011.
11. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, pages 192–206, 2008.
12. P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In *FC*, pages 325–343, 2009.
13. E. Boyle, K-M. Chung, and R. Pass. Large-scale secure computation: Multi-party computation for (parallel) ram programs. In *Crypto (2)*, pages 742–762, 2015.
14. E. Boyle, S. Goldwasser, and S. Tessaro. Communication locality in secure multiparty computation - how to run sublinear algorithms in a distributed setting. In *TCC*, pages 356–376, 2013.
15. R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai and T Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Eurocrypt*, pages 262–279, 2001.
16. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *STOC*, pages 11–19, 1988.
17. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *STOC*, pages 62–72, 1989.
18. T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. In *Latincrypt*, pages 40–58, 2015.
19. I. Damgård and Y. Ishai. Scalable secure multiparty computation. In *Crypto*, pages 501–520, 2006.
20. I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Eurocrypt*, pages 445–465, 2010.
21. I. Damgård, Y. Ishai, M. Krøigaard, J. Nielsen, A. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Crypto*, pages 241–261, 2008.
22. I. Damgård, J. Kilian, and L. Salvail. On the (im) possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Eurocrypt*, pages 56–73. Springer, 1999.

23. I. Damgård and J. Nielsen. Scalable and unconditionally secure multiparty computation. In *Crypto*, pages 572–590, 2007.
24. I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Crypto*, pages 643–662, 2012.
25. V. Dani, V. King, M. Movahedi, and J. Saia. Brief announcement: breaking the $o(nm)$ bit barrier, secure multiparty computation with a static adversary. In *PODC*, pages 227–228, 2012.
26. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. In *FOCS*, pages 36–45, 1990.
27. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *Crypto*, pages 205–210, 1983.
28. M. Fitzi, M. Franklin, J.A. Garay, and H.V. Simhadri. Towards optimal and efficient perfectly secure message transmission. In *TCC*, pages 311–322, 2007.
29. Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *FOCS*, pages 658–668, 2007.
30. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
31. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
32. O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems. *J. ACM*, 38(3), pages 691–729, 1991.
33. O. Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In *Crypto*, pages 73–86, 1988.
34. S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.
35. I. Haitner. Semi-honest to malicious oblivious transfer—the black-box way. In *TCC*, pages 412–426, 2008.
36. D. Harnik, Y. Ishai, and E. Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Crypto*, pages 284–302, 2007.
37. D. Harnik, Y. Ishai, E. Kushilevitz, and J. Nielsen. OT-Combiners via Secure Computation. In *TCC*, pages 393–411, 2008.
38. D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In *Eurocrypt*, pages 96–113, 2005.
39. M. Hirt, C. Lucas, and U. Maurer. A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In *Crypto (2)*, pages 203–219, 2013.
40. Y. Huang, J. Katz, and D. Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *Crypto (2)*, pages 18–35, 2013.
41. Y. Huang, J. Katz, V. Kolesnikov, R. Kumaresan, and A. Malozemoff. Amortizing garbled circuits. In *Crypto (2)*, 2014.
42. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61, 1989.
43. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Crypto*, pages 145–161, 2003.
44. Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Crypto*, pages 483–500, 2006.
45. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *Crypto*, pages 572–591, 2008.

46. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
47. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, pages 486–498, 2008.
48. T Kovári, V Sós and P Turán. On a problem of K. Zarankiewicz. *Colloquium Mathematicae* 3(1), pages 50–57, 1954.
49. E. Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421, 1989.
50. E. Larraia, E. Orsini, and N.P. Smart. Dishonest majority multi-party computation for binary circuits. In *Crypto (2)*, pages 495–512, 2014.
51. Y. Lindell, B. Pinkas, N.P. Smart, and A. Yanai. Efficient constant round multi-party computation combining bmr and spdz. In *Crypto (2)*, pages 319–338, 2015.
52. Y. Lindell and B. Riva. Cut-and-choose yao-based two-party computation with low cost in the online/offline and batch settings. In *Crypto*, pages 476–494, 2014.
53. Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. *Lecture Notes in Computer Science*, pages 1–17. Springer, August 2013.
54. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Eurocrypt*, pages 52–78, 2007.
55. H. Maji, M. Prabhakaran, and M. Rosulek. A zero-one law for cryptographic complexity with respect to computational UC security. In *Crypto*, pages 595–612, 2010.
56. R. Meier, B. Przydatek, and J. Wullschleger. Robuster combiners for oblivious transfer. In *TCC*, pages 404–418, 2007.
57. P. Mohassel and B. Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. In *Crypto (2)*, pages 36–53, 2013.
58. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
59. J. Nielsen, P. Nordholt, C. Orlandi, and S. Burra. A new approach to practical active-secure two-party computation. In *Crypto*, pages 681–700, 2012.
60. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *Crypto*, pages 554–571, 2008.
61. Michael Rabin. How to exchange secrets by oblivious transfer. 1981.
62. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC*, pages 73–85, 1989.
63. S. Wolf and J. Wullschleger. Oblivious transfer is symmetric. In *Eurocrypt*, pages 222–232, 2006.
64. W. Jürg Wullschleger. Oblivious-Transfer Amplification. In *Eurocrypt*, pages 555–572, 2007.
65. A. C.-C. Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.
66. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *Eurocrypt*, pages 220–250, 2015.
67. M. Zamani, M. Movahedi, and J. Saia. Millions of millionaires: Multiparty computation in large networks. In *ePrint 2014/149*.

Appendix A Introduction—continued

Remark on the generality of an OT infrastructure. Consider the following candidate for an infrastructure. Suppose there is a channel between a pair of parties

that allows them to securely evaluate any function. It is not hard to see that such a channel is equivalent to an OT channel. Since OT is complete for secure computation, one can apply the results of [45,46] to use the OT channel to implement a secure evaluation channel. The same argument extends trivially to channels that implement any 2-party primitive that is complete for secure computation [55,5]. In the other direction, one can use a secure evaluation channel to trivially implement OT channels. It is easy to see that the above argument also applies to the setting where a *set* of parties have a secure evaluation channel. Such a channel is equivalent to an OT graph where parties in the set have pairwise OT channels with everyone in the set.

Remark on the assumption of full network of secure channels. Secure channels between two parties can be implemented either via non-interactive key exchange and hybrid encryption or via a physical assumption. We emphasize that the one-time setup cost of emulating a secure channel (say via Diffie-Hellman key exchange) is much lower than the one-time setup cost of emulating an OT channel (that allows unbounded OT calls) via an OT protocol (even with OT extension). Furthermore, our assumption on secure channels is exactly the same as the assumptions in [46,33,45] who show that secure computation reduces to OT under information-theoretic reductions.

Appendix B Proofs for Lower Bound

Theorem 1. *Let $n \geq 2$ and $n/2 \leq t < n$, and let G be an n party network such that $G \subseteq \Lambda_{n-t}^{2t-n}$, with $P_1 \in V_A$ and $P_2 \in V_B$. Any t -secure OT protocol for G with P_1 as the sender and P_2 as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A,B} G_{\text{Kus}}$ with A' as the sender and B' as the receiver.*

Proof. Let G be an n party network with $G = (V, E)$ such that $G \subseteq \Lambda_{n-t}^{2t-n}$. Then, we may write $V_U = V_A \dot{\cup} V_S \dot{\cup} V_B$, where $|V_A| = |V_B| = n - t$ and $|V_S| = 2t - n$, with $P_1 \in V_A$ and $P_2 \in V_B$ and $E(V_A, V_B) = \emptyset$, where $E(V_A, V_B)$ represents the set of edges with one endpoint in V_A and the other in V_B .

Let Π be a t -secure OT protocol for G with P_1 as the sender and P_2 as the receiver. If $t > n/2$, then we can use Π to construct a 2-secure OT protocol Π' for any three party network G' with $G' \simeq_{A',B'} G_{\text{HIK}}$ with A' as the sender and B' as the receiver below. Combining this with Lemma 3, the conclusion follows. We describe the construction of Π' below. If $t = n/2$, then we can use Π to construct a 1-secure OT protocol Π'' for any two party network G'' with $G'' \simeq_{A'',B''} G_{\text{CK}}$ with A'' as the sender and B'' as the receiver. The construction of Π'' is exactly the same as that of Π' and hence we omit its description here.

In protocol Π' , party A' simulates the parties of component V_A , party C' simulates the parties of component V_S , and party B' simulates the parties of component V_B . Executions of 1-secure OT protocols between parties of the same component are handled locally and executions of 1-secure OT protocols between parties in different components is handled as follows:

- If the parties are in components V_A and V_S , then executions of 1-secure OT protocols between the parties are carried out using the OT edge $\{A', C'\}$ in the network G' .
- If the parties are in components V_B and V_S , then executions of 1-secure OT protocols between the parties are carried out using the OT edge $\{B', C'\}$ in the network G' .

Since $G \subseteq \Lambda_{n-t}^{2t-n}$, there are no executions of 1-secure OT protocols between parties in components V_A and V_B in the protocol Π .

Correctness of Π' is obvious. We now prove 2-security of Π' . Intuitively, since Π is t -secure, in particular, it is secure against corruptions of parties $V_A \cup V_S$ or the parties $V_B \cup V_S$. Consequently protocol Π' is secure against corruptions $\{A', C'\}$ or $\{B', C'\}$ and hence Π' is a 2-secure protocol.

Formally, we prove this as follows. Since Π is t -secure, there exists a simulator \mathcal{S} such that for every PPT t -threshold adversary \mathcal{A} , $\left\{ \text{REAL}_{\Pi, \mathcal{A}}^{(x_0, x_1), b, \perp, \dots, \perp} \right\} \equiv \left\{ \text{IDEAL}_{\Pi, \mathcal{A}, \mathcal{S}}^{(x_0, x_1), b, \perp, \dots, \perp} \right\}$, where $x_0, x_1 \in \{0, 1\}^m$ and $b \in \{0, 1\}$. The simulator \mathcal{S}' for the protocol Π' behaves exactly the same as \mathcal{S} while simulating the parties of component V_A as A' , those of component V_B as B' and those of component V_S as C' . It is easy to see that for any 2-threshold adversary \mathcal{A}' , namely, one which corrupts $\{A', C'\}$ or the one which corrupts $\{B', C'\}$, $\left\{ \text{REAL}_{\Pi', \mathcal{A}'}^{(x_0, x_1), b, \perp} \right\} \equiv \left\{ \text{IDEAL}_{\Pi', \mathcal{A}', \mathcal{S}'}^{(x_0, x_1), b, \perp} \right\}$ since Π is secure against corruptions of parties $V_A \cup V_S$ or the parties $V_B \cup V_S$. \square

Appendix C Proofs for building blocks

In this section we provide the remaining proofs.

Lemma 4. *Protocol 1 is an efficient t -secure OT protocol for a network $G \simeq_{A, B} G_{\text{claw}}^t$ with A as the sender and B as the receiver.*

Proof (Lemma 4). Let \mathcal{A} be a t -threshold adversary which corrupts parties T , $|T| \leq t$. We will construct a simulator \mathcal{S} which plays the role of the uncorrupted parties. If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so \mathcal{S} can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \emptyset$ then \mathcal{S} chooses arbitrary inputs x_0, x_1, b and runs the protocol, invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since corrupted parties only learn secret shares of independently random values, the view of the adversary is independent of the choice of x_0, x_1, b and is identical to the real world.

Otherwise, we have that the corrupted parties T include exactly one of A, B . If $A \in T$ but $B \notin T$, then \mathcal{S} chooses arbitrary input b and runs the protocol, invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since the OT simulator does not reveal the input c , and since the adversary only learns the direct sum of b with the random bit c , the view of the adversary is identical regardless of the value of b and in particular is identical to the real world.

Finally we have the case $B \in T, A \notin T$. Here the simulator \mathcal{S} is given the output value x_b . \mathcal{S} runs the protocol with (x_b, x_b) as the input to A , again invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since $|T| \leq t$ and $B \in T$, at most $t - 1$ of the t parties P_3, \dots, P_{t+2} are corrupted. Consequently the adversary observes at most $t - 1$ shares of the random one-time pads r_0, r_1 , so by the security of t -out-of- t secret sharing, conditioned on the remaining shares being hidden, the distribution of the observed shares is independent of r_0, r_1 . The adversary learns the shares of r_c , but by the security of the OT channels, the view of the adversary in step 3 and onward is independent of the remaining shares of r_{1-c} and consequently is independent of the choice of r_{1-c} . Consequently the view of the adversary in step 3 and onward is independent of r_{1-c} . In step 2, the adversary sees $y_b = x_b \oplus r_c$ and $y_{1-b} = x_{1-b} \oplus r_{1-c}$, so by the security of the one-time pad, the view of the adversary is independent of x_{1-b} . Consequently the overall view of the adversary is identical in the real and ideal worlds. \square

Lemma 5. *Protocol 2 is an efficient t -secure OT protocol for a network $G \simeq_{A,B} G_{\text{clique}}^t$ with A as the sender and B as the receiver.*

Proof (Lemma 5). Let \mathcal{A} be a t -threshold adversary which corrupts parties T , $|T| \leq t$. We will construct a simulator \mathcal{S} which plays the role of the uncorrupted parties. As above, if $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so \mathcal{S} can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \emptyset$ then \mathcal{S} chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend at all on the inputs are on point-to-point channels between A and B , the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that the corrupted parties T include exactly one of A, B . If $A \in T$ but $B \notin T$, then \mathcal{S} chooses arbitrary input b and runs the protocol, invoking the MPC simulator for the protocol in step 1 (the existence of this simulator follows from Lemma 2). Since at least one of the parties P_3, \dots, P_{t+2} is uncorrupted, the security of the MPC protocol implies that the view of the adversary is independent of the uncorrupted parties' shares s^i and c^i , and so by the security of the secret sharing scheme is independent of the value of the bit c . The only message received by the adversary which depends on b is the bit $b' = b \oplus c$, so it follows that the view of the adversary is independent of the bit b and therefore is identical in the real and ideal worlds.

Finally we have the case $B \in T, A \notin T$. Here the simulator \mathcal{S} is given the output value x_b . \mathcal{S} runs the protocol with (x_b, x_b) as the input to A , again invoking the MPC simulator for the protocol in step 1. Since at least one of the parties P_3, \dots, P_{t+2} is uncorrupted, the security of the MPC protocol implies that the view of the adversary is independent of the uncorrupted parties' shares r_0^i and r_1^i , so by the security of t -out-of- t secret sharing, conditioned on the remaining shares being hidden, the distribution of the observed shares is independent of r_0, r_1 . The adversary learns the shares of r_c , but by the security of the OT channels, the view of the adversary through step 4 is independent of the remaining shares of r_{1-c} and consequently is independent of the choice of r_{1-c} . In step 5,

the adversary sees $y_b = x_b \oplus r_c$ and $y_{1-b} = x_{1-b} \oplus r_{1-c}$, so by the security of the one-time pad, the view of the adversary is independent of x_{1-b} . Consequently the overall view of the adversary is identical in the real and ideal worlds. \square

Appendix D Efficiency analysis of Protocol 5

Theorem 6. *Protocol 5 runs in quasi-polynomial time. Assuming one-way functions, we can obtain a computationally secure protocol which runs in polynomial time using computationally secure cascading (Lemma D).*

Proof. Each iteration of step 1 decreases the length of a path between any pair of vertices from ℓ to $\lceil \ell + 1 \rceil / 2$. Consequently, after $O(\log n)$ iterations the graph will consist of a collection of disjoint cliques, and the protocol will move on to the next step. By Lemma 6 (Cascading), if each iteration can be performed in time at most T assuming the augmented graph, then the full cascaded protocol runs in time at most $T^{O(\log n)}$. Since $T = \text{poly}(n)$ and each other step of the protocol is efficient, this implies that Protocol 5 runs in quasi-polynomial time.

Replacing the cascading of step 1 with the more efficient but computationally secure cascading of Lemma , we have the cascaded protocol runs in time $O(T \text{poly}(\lambda) \cdot \log n)$. Since each other step of the protocol is efficient, this implies that assuming one-way functions, we have a computationally-secure version of Protocol 5 that runs in quasi-polynomial time. \square

Appendix E Proofs for Protocols

Theorem 3. *Consider an n -party network G , which contains A and B as two of the parties. Let $t = n - 2$. Protocol 6 is an efficient t -secure OT protocol with A as the sender and B as the receiver if and only if G is $(2, A, B)$ -unsplittable.*

Proof (Theorem 3). The “only if” part of theorem has been proved by virtue of the lower bound proven in Theorem 1 with $t = n - 2$. We now prove the “if” part. Let \mathcal{A} be a t -threshold adversary which corrupts parties T , $|T| \leq t = n - 2$. If A and B are both corrupt then the uncorrupted parties receive no input, so the simulator \mathcal{S} can perfectly simulate the uncorrupted parties.

Otherwise we have that at least one of A or B is uncorrupted. We first show how to simulate step 1. Since G is $(2, A, B)$ -unsplittable, for each pair of parties P_i, P_j considered in step 1, we have that $G_{i,j}$ is $(2, A, B)$ -unsplittable. Since $\{P_i, P_j\} \notin E$, this implies that some vertex among A, B, P_3 , and P_4 has degree at least 2 in $G_{i,j}$. In particular, we have $G_{i,j} \not\subseteq_{P_3, P_4} \Lambda_2^0$, where Λ_2^0 is labeled so that P_3 and P_4 are in separate components. Consequently, by Theorem 2, we securely obtain OT correlations between P_3 and P_4 whenever at most two of the parties A, B, P_3, P_4 are corrupt.

We have that A and B are not both corrupt. Therefore, if P_3 and P_4 are not both corrupt, then at most two of the parties A, B, P_3, P_4 are corrupt. Consequently we securely obtain OT correlations between P_3 and P_4 . On the other

hand, P_3 and P_4 are both corrupted, then there is nothing to prove. More formally, there is a simulator which can perfectly simulate the invocation of Protocol 5 in step 1a by executing the protocol of the honest parties, since the honest parties receive no input. Consequently we securely obtain OT correlations between the corrupted parties P_3 and P_4 . In both cases, by Lemma 6, we have an OT simulator which can simulate subsequent invocations of the OT channel between P_3 and P_4 .

Therefore, by the end of step 1 we t -securely obtained an OT channel between every pair of parties P_i, P_j for $i, j \geq 3$. Then by Lemma 5, the invocation of Protocol 2 in step 2 is a t -secure OT protocol with A as the sender and B as the receiver, as desired. \square

Lemma 12. *If G is $(n-t, A, B)$ -unsplittable, then G' is a $(2, A, B)$ -unsplittable network on $n' = \binom{n-2}{n-t-1} + 2$ vertices, where G' is the network produced in Protocol 8.*

Proof (Lemma 12). We prove the contrapositive. Assume that $G' \subseteq_{A,B} \Lambda_2^{n'-2}$. Then there exist vertices $X, Y \in S_{k-1}$ such that there are no edges in G' between any of the parties in $\{A, X\}$ and any of the parties in $\{B, Y\}$. In particular, $X \cap Y = \emptyset$, since otherwise $\{X, Y\}$ would be an edge of G' . This implies that we have $2k = 2(n-t)$ parties $\{A, B\} \cup X \cup Y$ such that there are no edges in G from the $n-t$ parties $\{A\} \cup X$ to any of the $n-t$ parties $\{B\} \cup Y$. By definition, this means that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$, which is a contradiction. \square

Theorem 5. *Consider an n -party network G , which contains A and B as two of the parties. Let $t \geq n/2$. Protocol 8 is a t -secure OT protocol with A as the sender and B as the receiver if and only if G is $(n-t, A, B)$ -unsplittable. The protocol is efficient for $t = n - \mathcal{O}(1)$.*

Proof (Theorem 5). As before, we only need to prove the “if” part. Let \mathcal{A} be a t -threshold adversary which corrupts parties T , $|T| \leq t$. If A and B are both corrupt, then the honest parties have no input, so the simulator \mathcal{S} can perfectly simulate the uncorrupted parties. If A and B are both honest, then \mathcal{S} chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend at all on the inputs are on point-to-point channels between A and B , the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that at most $t-1$ of the parties P_3, \dots, P_n are corrupt and that either A or B is honest. In particular, for $k = n-t$, there are at least $k-1$ uncorrupted parties among P_3, \dots, P_n . Consequently, S_{k-1} contains some set X consisting only of honest parties. We treat a party X in G' as honest if all constituent parties $P_i \in X$ are honest. Since A and B are not both corrupt, we have that G' contains at least two honest parties. By Lemma 12, G' is $(2, A, B)$ -unsplittable. Consequently by Theorem 3 there is a simulator \mathcal{S}' which simulates the roles of the honest parties in the invocation of Protocol 6 on G' in step 1. We define a simulator \mathcal{S} for Protocol 8 which behaves exactly as an honest party for communication within each party $X \in S_{k-1}$ and invokes \mathcal{S}' for any

communication between parties in G' . The behavior of this protocol is identical to the behavior of \mathcal{S}' . Hence, by the correctness of simulator \mathcal{S}' , we have that the view of the adversary is identical in the real and ideal worlds. \square

Lemma 11. *Given graph $G = (V, E)$ and a vertex i , let G_i be the induced graph on the $n - 1$ vertices $V \setminus \{i\}$. If G is $(n - t, A, B)$ -unsplittable, then G_i is also $(n - t, A, B)$ -unsplittable.*

Proof. We will prove the contrapositive. Suppose that $G_i \subseteq_{A,B} \Lambda_{n-t}^{2t-n-1}$. This means there exists a partition of the vertex set of G_i as $V \setminus \{i\} = V_A \dot{\cup} V_S \dot{\cup} V_B$ where $A \in V_A$, $B \in V_B$, $|V_A| = |V_B| = n - t$ and $|V_S| = 2t - n - 1$. Now, we note that there exists a partition of the vertex set of G as $V = V_A \dot{\cup} V'_S \dot{\cup} V_B$ where $A \in V_A$, $B \in V_B$, $V'_S = V_S \cup \{i\}$ and hence $|V_A| = |V_B| = n - t$ and $|V'_S| = 2t - n$. This implies that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$, which is a contradiction. \square

Corollary 1. *Let G be an n -party network. For $t \geq n/2$, we can t -securely generate OT correlations between all pairs of parties (thus, completing the OT network) if and only if the G is $(n - t)$ -unsplittable.*

Proof. If G is $(n - t)$ -unsplittable, then for all pairs of vertices A, B , G is $(n - t, A, B)$ -unsplittable and hence from Theorem 4, we can generate OT correlations between A and B .

Conversely, if $G \subseteq \Lambda_n^{2t-n}$, then choosing vertices $A \in V_A$ and $B \in V_B$, we have that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$ and hence Theorem 4 rules out the existence of such a protocol. Hence, there exist a pair of vertices between whom we cannot t -securely generate OT correlations. This completes the proof. \square

Appendix F Bounding the number of edges in $\approx \frac{n}{2}$ -unsplittable graphs.

In this section we discuss the minimum number of edges in a graph which is $(n - t)$ -unsplittable for $t = \lfloor (n + 1)/2 \rfloor$. We show that the minimum edge count is $n/2$ if n is even and $t = n/2$, and $(n + 3)/2$ if n is odd and $t = (n + 1)/2$. These bounds give the minimum number of OT channels required to obtain t -secure MPC among n parties in a network for $t = \lfloor (n + 1)/2 \rfloor$. They constitute the first nontrivial cases, since no OT channels are needed in the case of an honest majority.

Theorem 7. *Let n be even and $t = n/2$. Then any $(n - t)$ -unsplittable graph must contain at least t edges. This bound is tight.*

Proof. To show that the bound is tight, note that the t -claw graph (Figure 4(a)) (or any graph with a tree on $t + 1$ vertices and no other edges) has t edges and contains a connected component consisting of $t + 1 = n/2 + 1$ vertices, so it is $(n/2)$ -unsplittable. We now show that every graph with fewer edges can be split.

Let G be a graph containing $t - 1$ edges. Let m be the number of connected components of G , let C_1, C_2, \dots, C_m be the components in nonincreasing order of size, and let $\alpha_i = |C_i|$, so that $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_m$ and $\sum_{i=1}^m \alpha_i = n$. G is $(n - t)$ -splittable if and only if there is some subset $I \subset [m]$ such that $\sum_{i \in I} \alpha_i = n/2$, that is, some subset of the values α_i sum to $n/2$.

For all natural numbers x , let $a_x = |\{i : \alpha_i \geq x\}|$ denote the number of *connected components* with size at least x , and let $b_x = |\{i : \alpha_i < x\}|$ denote the number of *vertices* in connected components with size smaller than x . Note that a_x counts the components of certain sizes, while b_x counts the vertices contained in components of certain sizes. Since a component of size s must contain at least $s - 1$ edges, for any $x > 1$ we have that $a_x \leq (t - 1)/(x - 1) < n/(2x - 2)$. In particular, we have that $a_2 < n/2$, $a_3 < n/4$, and $a_4 < n/6$.

Since G has at most $n/2 - 1$ edges, at least two vertices have degree zero, so $\alpha_1 = \alpha_2 = 1$. Consequently $b_2 \geq 2$. A set of c components containing at most $t - 1$ edges can include at most $c + t - 1$ vertices. Consequently, for any $x > 1$, we have that

$$b_x \geq n - (a_x + t - 1) = \frac{n}{2} + 1 - a_x.$$

In particular, we have that $b_3 > \frac{n}{4} + 1$ and $b_4 > \frac{n}{3} + 1$.

Since $\alpha_1 \geq \alpha_2 \geq \alpha_3$, the third largest component has size $\alpha_3 \leq n/3$. There are only $n/2 - 1$ edges, so $\alpha_i \leq n/2$ for all i .

Consider the following greedy algorithm. Initially let $S_2 = C_2$. For i from 3 to n , if $|S_{i-1} \cup C_i| \leq n/2$ then set $S_i = S_{i-1} \cup C_i$, and otherwise set $S_i = S_{i-1}$. We show that at the end of this loop, the set S_m will always have size $n/2$.

Since $|C_m| \leq n/2$, the sum of the sizes of the other components is at least $n/2$, so if $|S_{i+1} \cup C_i| \leq n/2$ for every i considered in the loop then the loop must terminate with $|S_1| = n/2$. Otherwise there is some i such that $|S_{i-1} \cup C_i| > n/2$. Choose the last iteration i for which this is true. Since $i \geq 3$, we must have that $\alpha_i \leq \alpha_3 \leq n/3$. If $\alpha_i > 3$, then since $n/2 - |S_i| \leq \alpha_i - 1 \leq n/3 - 1$ and $b_4 > \frac{n}{3} + 1$, we reach a contradiction with the assumption that i is the last such iteration, since there are enough vertices in components C_{i+1}, \dots, C_m to make some subsequent $|S_{j-1} \cup C_j| > n/2$. Otherwise $\alpha_i \leq 3$, so since $n/2 - |S_i| \leq \alpha_i - 1 \leq 2$, so since $b_2 \geq 2$, we must have that $|S_m| = n/2$.

Consequently at the end of the loop we always have that $|S_m| = n/2$. Since S_m consists only of entire connected components, S_m consists of $n/2$ vertices with no edges to the rest of the graph, so $G \subset \Lambda_{n/2}^0$ cannot be $(n/2)$ -unsplittable. \square

Theorem 8. *Let n be odd and $t = (n+1)/2$. Then any $(n-t)$ -unsplittable graph must contain at least $t + 1$ edges. This bound is tight.*

Proof. To show that the bound is tight, note that the $(t+1)$ -edge graph consisting of a cycle on $t + 1$ vertices is $(n - t)$ -unsplittable, since removing any vertex still leaves a connected component of size $t - 1$. We now show that every graph with fewer edges can be split.

Let $G = (V, E)$ be a graph containing $|E| = t$ edges. If the maximum degree is 1, then G contains at most $n/2 < t$ edges, a contradiction. Let v be a vertex

of maximal degree ≥ 2 . The induced subgraph on vertex set $V \setminus \{v\}$ contains $n - 1$ vertices and at most $t - 2 < (n - 1)/2$ edges. By Theorem 7 it cannot be $((n - 1)/2)$ -unsplittable, so G cannot be $((n - 1)/2)$ -unsplittable either. \square