On the Correlation Complexity of MPC with Cheater Identification

Nicholas Brandt[1], Sven Maier[2], Tobias Müller[3], and Jörn Müller-Quade[4]

[1] ETH Zurich, Zurich, Switzerland, `nicholas.brandt@inf.ethz.ch`
[2] CNRS, IRIF, Université de Paris, France, `sven.maier@irif.fr`
[3] `muellertobias@outlook.de`
[4] Karlsruhe Institute of Technology, Germany, `joern.mueller-quade@kit.edu`

Abstract. Composable protocols for Multi-Party Computation that provide security with Identifiable Abort against a dishonest majority require some form of setup, e.g. correlated randomness among the parties. While this is a very useful model, it has the downside that the setup's randomness must be *programmable*, otherwise security becomes provably impossible. Since programmability is more realistic for smaller setups (in terms of number of parties), it is crucial to minimize the correlation complexity (degree of correlation) of the setup's randomness.

We give a tight tradeoff between the correlation complexity β and the corruption threshold t. Our bounds are strong in that β-wise correlation is sufficient for statistical security while $\beta - 1$-wise correlation is insufficient even for computational security. In particular, for strong security, i.e., $t < n$, full n-wise correlation is necessary. However, for any constant fraction of honest parties, we provide a protocol with *constant* correlation complexity which tightens the gap between the theoretical model and the setup's implementation in the real world. In contrast, previous state-of-the-art protocols require full n-wise correlation regardless of t.

1 Introduction

Secure Multi-Party Computation (MPC) is a powerful notion that allows multiple mutually distrustful parties to perform a joint computation that—loosely speaking—ensures the privacy of the inputs and the correctness of the output. The currently strongest security notion—that is not ruled out by some impossibility result [16]—is called security with Identifiable Abort (IA) [30]. It allows an adversary to abort the protocol (this is unavoidable) but then the honest parties can identify the common identity of at least one malicious party. This acts as a deterrent against cheating by coupling cheater identification to some form of penalty mechanism. This is especially useful in the context of blockchains where one could require all parties to initially commit to some coins s.t. an identified cheater's coins are redistributed to the other parties or the cheater's coins are rendered void by publishing the evidence of cheating.

* Work done while the first author was supported by ERC Project PREP-CRYPTO 724307, the second author was at the Karlsruhe Institute of Technology, Germany, and the third author was at the FZI Research Center for Information Technology, Germany.

In the dishonest majority settings, $t \geq n/2$, protocols such as the one of Ishai, Ostrovsky, and Zikas [30] that achieve IA require a setup that distributes correlation randomness to each party in the protocol. In fact, for $t \geq n/2$, a setup is provably necessary for general MPC protocols that can be composed arbitrarily, e.g. in the Universal Composability (UC) framework [11]. Moreover, for the security proof to work the setup needs to be *programmable* to realize certain functionalities such as commitments [12]. That is, the setup information may not leak directly to the environment, instead, in the security proof the simulator must be able to embed a trapdoor into the setup information to extract or equivocate the committed message. Indeed, if the setup is *global*, the setup information leaks directly to the environment, then many functionalities become *provably impossible* [37, 13]. In particular, if in practice the setup information is extracted from some public source, like stock market data, then the security guarantee provided by the ID-MPC protocol is void. This leaves the option to generate the correlated randomness via some physical means, like noisy or quantum channels, or secure hardware assumptions. However, for such a means of generating randomness the correlation complexity (CC) is the most important parameter.
As shown in [30] the correlated randomness setup for n-parties suffices to statistically securely realize any other functionality (or setup) of cardinality n (with n-participants). Therefore, we equate the correlation complexity (see Definition 5) with the minimal complete cardinality (MCC)[1] as introduced by Fitzi et al. [23].

For $t < n/2$ pairwise correlation (even pairwise channels) suffice [38, 5], while for $t \geq n/2$ protocols like [30] are quite conservative in that they require (maximal) n-wise correlation; even for $t = n/2$. Our work closes this gap between $t = n/2$ and $t = n - 1$ by answering the question:

"What is the correlation complexity of MPC with a dishonest majority?"

We settle this question with tight bounds for the correlation complexity $\beta \approx 2n/(n-t)$ depending on the max. number of corrupted parties t. See Fig. 1 for an exemplary overview.

While theoretically interesting,[2] our results offer also two practical insights:

- If one requires maximal security $t < n$, then n-wise correlation is necessary, i.e., the CC is $\beta = n$. Hence protocols like [30] are optimal w.r.t. the CC.
- If one is willing to accept any constant fraction of malicious parties $t \leq (1-\varepsilon)n$ for any $\varepsilon > 0$, then the CC is only constant $\beta \approx 2/\varepsilon$.

Especially the latter case has practical implications. Due to the aforementioned *provable impossibility* of composable MPC with a global setup, the setups must be realized by non-cryptographic means such as trusted hardware [27, 39] or noisy/quantum channels [21, 19]. There, the CC (the number of setup participants) is a critical parameter. To illustrate this, consider the following example:

[1] Throughout the paper, we require a setup among each subset of parties of size β.

[2] To our knowledge this is the first full characterization of Identifiable Abort in the dishonest majority setting.

Max. malicious parties t	Max. supported parties n	CC / MCC β
$n-1$	$\mathrm{poly}(\lambda)$	n
$n-2$	$\mathrm{poly}(\lambda)$	$n-2$
$n-c \leq n-3$	$\Theta(\ln \lambda)$	$\approx 2n/c$
$n-\Theta(\ln n)$	$\mathcal{O}(\ln(\lambda)\ln\ln(\lambda)/\ln\ln\ln(\lambda))$	$\Theta(n/\ln n)$
$n-\Theta(\sqrt{n})$	$\mathcal{O}(\ln^2(\lambda)/\ln^2\ln(\lambda))$	$\Theta(\sqrt{n})$
$n-\Theta(n/\ln n)$	$\mathcal{O}(\exp\sqrt{\ln\lambda})$	$\Theta(\ln n)$
$\Theta(n)$	$\mathrm{poly}(\lambda)$	$\Theta(1)$
$(n+1)/2$	$\mathrm{poly}(\lambda)$	3
$\leq n/2^{(*)}$	$\mathrm{poly}(\lambda)$	2

Table 1: Exemplary overview of the correlation complexity (CC) / minimal complete cardinality (MCC) β and respective supported number of parties n vs. malicious parties t for UC-secure ID-MPC given broadcast. The limitation of the overall number of parties is only to achieve polynomial-time protocols, for more parties the protocols remain correct and secure but require the parties to have superpolynomial runtime. The case $(*)$ also covers an honest majority of parties treated in early works [5, 38].

Suppose a group of people can generate correlated randomness via some trusted hardware in their smartphones while being online simultaneously. However, the runtime of this supposed setup computation is exponential in the number of parties involved. In this scenario, if a protocol relies on a single setup of cardinality n, then all parties must be online fogether or an exponential time. In contrast, if a protocol could be based on (polynomially) many setup instances between a constant number of people (as is the case for our protocols for any $t \leq (1-\varepsilon)n$), then a) being online at the same time as a constant number of parties much more realistic and b) the runtime is only polynomial in the number of people.

The above example showcases that our results are particularly interesting for applications in which mobile clients (which are not always online) perform decentralized operations and store a common state on some form of blockchain (which allows for monetary penalties for cheating).

1.1 Contributions & Techniques

Correlation complexity / ID-MPC from small setups. Due to the completeness of the correlated randomness setup [30], we can substitute any setup functionality of some cardinality by the correlated randomness functionality of the same cardinality. To minimize the correlation complexity we concentrate on constructing n-party ID-MPC from the smallest possible (arbitrary) setups. In other words, we can answer the question of the CC by determining the minimal complete cardinality (MCC)[3] in the sense of [23], i.e., the number of participants

[3] As a side note we generalize the notion of the minimal complete cardinality (MCC) from [23] to the setting where the number of parties varies in the security parameter λ. This was not captured by the original definition of MCC in [23] and—to the best of our knowledge—not formally addressed in previous literature.

of a setup functionality. As a sidenote, we deal with some definitorial issues in Appendix B.1 that arise when the number of parties grows with the security parameter which is not the case in [23]. As our main result we establish tight bounds on the minimal complete cardinality β for general ID-MPC (given broadcast). We assume that each subset of parties (of cardinality β) has access to a setup functionality. Furthermore, we only require these setups to guarantee security with Identifiable Abort—unlike many other works which don't allow the setups to be aborted at all. For a formal description of our setting see Appendix B.2.

Theorem 1 (Correlation complexity bounds). *The correlation complexity for UC-secure Multi-Party Computation with Identifiable Abort (given broadcast) is* $\beta \coloneqq \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2) \approx 2n/(n-t)$ *where* n *is the overall number of parties and* t *is an upper bound on the number of malicious parties. In other words, for any* n*-party functionality with Identifiable Abort there exists a protocol that uses hybrid functionalities of cardinality* β *and broadcast, but there exists an* n*-party functionality with Identifiable Abort which cannot be realized by any protocol that uses hybrid functionalities of cardinality* $\beta - 1$ *and broadcast.*

Identification via conflicts. Towards our main result we formalize an intuitive mechanism for cheater identification that is also used in various other works [32, 28, 30, 3, 4, 42, 41] in different contexts. In our application, all parties maintain a global data structure, namely a graph with one vertex per party where each party can remove incident edges (we call missing edges "conflicts") but never add edges. Following Wan et al. [42] we call this structure "Trust Graph" (TG). With it, we provide a ruleset for its usage which we call *abort-respecting* (see Definition 8) that ensures that the Trust Graph exhibits certain useful properties: On an intuitive level, a disconnected subgraph corresponds to an aborted setup and vice-versa while a disconnected overall TG corresponds to the honest parties' ability to abort by identifying malicious parties.

Lemma 1 (Informal conflict reporting). *Any protocol that securely realizes an ideal functionality* $\mathcal{F}$ *in some hybrid model can be modified such that*

1. *all honest parties keep a (common) Trust Graph,*
2. *if the Trust Graph is disconnected, then all honest parties can identify the same malicious parties,*
3. *upon abort of the protocol its Trust Graph is disconnected,*
4. *after abort of any setup (hybrid functionality on some subset of parties[4]) in the protocol the corresponding subgraph of the Trust Graph becomes disconnected,*

and the modified protocol still securely realizes the same functionality.

As a consequence, the impossibility of any abort-respecting protocol implies the impossibility of any protocol. On the other hand, if some protocol for a given functionality exists, then so does an abort-respecting protocol. In consequence, we only need to consider abort-respecting protocols.

[4] Throughout the paper we assume that each subset of parties of the appropriate cardinality has access to a setup.

For the lower and the upper bound, we prove two complementing graph-theoretical lemmas that link the connectivity of the overall TG to the connectivity of its subgraphs. In a nutshell, a connected graph of cardinality n can have "many" disconnected subgraph of cardinality $\beta - 1$ but only "few" disconnected subgraphs of cardinality β.

For our lower bound we devise a strategy for the adversary such that it can abort many setups of cardinality $\beta - 1$ while the overall Trust Graph remains connected. Following the proof strategy of Canetti and Fischlin [12] we can show that against such an adversary any protocol for a commitment must violate either the hiding or the binding property. For our upper bound we know that any adversary can only abort "few" setups. Thus the honest parties can rely on some "guaranteed" setups to perform the protocol.

1.2 Related and Concurrent Work

There are many works that share common aspects with this paper, among others [23, 24, 32, 28, 42, 29, 30, 3, 4, 41]. Here we pick only the most closely related ones and describe their relation to this work.

- Fitzi et al. [23] initialize the study of the minimal complete cardinality (MCC)—the cardinality of the smallest setup (least number of participants) that suffices to securely realize any n-party functionality.
- Ishai, Ostrovsky, and Seyalioglu [29] rule out pairwise setups plus broadcast for statistically secure ID-MPC.
- Ishai, Ostrovsky, and Zikas [30] formally define Identifiable Abort (IA), introduce the Correlated-Randomness model for IA and give a computationally secure construction from any adaptively secure OT protocol.
- Wan et al. [42] use the almost identical idea of maintaining a "Trust Graph" (TG) in the context of *constructing* Byzantine broadcast (BB) while we assume a broadcast to construct general ID-MPC. Specifically, Wan et al. [42] give lower bounds for the round-complexity of BB. From the different applications arises the slight difference in the two concepts, in [42] each party maintains its own copy of the TG whereas in our work it is crucial that all parties have a common view of the TG. Nevertheless, we use very similar graph properties as Wan et al. [42]. Their idea is to limit the distance that information can travel within a graph in a given number of round, if the graph's diameter is too large, then a sender's message may not be able to reach all other parties. Their upper bound for the TG's diameter d matches our tight bound for the CC, i.e., $\beta = \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2 = d+1$ where $t \geq n-2$ is a lower bound on the number of honest parties.

  We think there are interesting connections[5] between our paper and [42]. While [42] assumes pairwise channels, we assume a full broadcast. We maintain the view that both [42] and our paper can be generalized along the dimension of the setup size, i.e., assuming a partial broadcast of size $2 \leq k \leq n$.

[5] In particular, Claim 3.1 in [42] and our Lemma 3 share the same core idea but are stated in different terms with different applications in mind. Also, Figure 1 in [42] essentially matches our Fig. 1.

Regarding the round-complexity of BB, it seems that the round-complexity decreases as k increases because the sender's message travels farther in each round. Regarding the correlation complexity, it seems that the CC increases as k increases because for larger partial broadcasts reaching a consensus on a identified cheater seems easier.

- Simkin, Siniscalchi, and Yakoubov [41] essentially study the same question as our paper. They give a weaker upper bound $\beta \leq t + 2 \leq n - 2$ of the CC/MCC, although in the stand-alone model whereas our result holds in the UC framework.
  They construct n-party MPC from correlation of degree $n - 1$ and broadcast. For this reason their work supports polynomially many parties $n \in \text{poly}(\lambda)$ only for $n - t \in \Theta(1)$. For larger expansions the supported number of parties drops rapidly since the overall runtime grows exponentially in the number of recursive applications of the protocol. This is not the case for our work; see Appendix A for a discussion.
  Their approach uses an new form of identifiable secret-sharing with public and private shares. There, one party P is chosen and the remaining $n - 1$ parties obtain correlated randomness, i.e., secret-shares of their randomness, from the setup oracle. Then the parties send their shares to the excluded party P who reconstructs its randomness. If reconstruction fails to due faulty shares sent by malicious parties, then party P detects whose shares where faulty and declares conflicts with these parties. These conflicts are then used in the next iteration. That is, conflicting parties do not obtain shares from the setup.
  It seems unclear how the approach of [41] could be generalized to setups of cardinality $n - 2$ without the disadvantageous recursion blowup.
- Finally, much work [7, 9, 8, 36, 18] has gone into reducing the necessary length of correlated randomness. This is highly relevant in practice. Nevertheless, to the best of our knowledge, in these works all parties need to participate in the correlation generation simultaneously. That is, although the overall length of the correlated randomness is short, the degree of the correlation is maximal—which is where our work steps in. We'd like to emphasize that our protocols are compatible with approaches to reduce the length of the correlated strings, and they can be used in conjuction to reduce both the length as well as the degree of correlation.

Our results subsume or improve upon all previously listed constructions and impossibilities in a unified way. For the minimal complete cardinality (MCC) it holds that:

- The lower bound of 3 from [29] for $t \geq 2n/3$ is raised to $\min(n, 5)$.
- The upper bound of $n-1$ from [41] reduced to the optimal $n-2$ for $t \leq n-2$.
- The upper bound of n from [30] is shown to be tight for $t = n - 1$.

These (and more) results are summarized in Table 2. In the following we use the short notation $F \rightsquigarrow_t \mathcal{F}$ for the fact that the ideal functionality $\mathcal{F}$ can be realized by some protocol in the F-hybrid model with up to t malicious parties (see Notation 8).

No.	Reference	Model	Result	Technique
1)	[29]	SA	$\{\mathcal{F}^2, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\not\rightsquigarrow}_{2n/3} \mathcal{F}^n_{\mathsf{SFE},f}$	Secret-Sharing
2)	[30]	UC, SA	$\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}} \overset{\text{stat}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{SFE},f}$	Setup+Commit+Prove
3)	[30]	UC, SA	$\{\pi_{\mathsf{OT}}, \mathcal{F}^n_{\mathsf{CRS}}\} \overset{\text{comp}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$	Setup+Commit+Prove
4)	[41]	SA	$\{\mathcal{F}^{n-1}_{\mathsf{Corr},\mathcal{D}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\rightsquigarrow}_{n-2} \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$	Secret-Sharing
5)	This work	UC	$\{\mathcal{F}^{\beta}_{\mathsf{Corr},\mathcal{D}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{SFE},f}$	Trust Graph
6)	This work	UC	$\{\mathcal{F}^{\beta-1}_{\mathsf{Corr},\mathcal{D}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{comp}}{\not\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{COM},1:1}$	Trust Graph

Table 2: Overview of related work on the foundations of Multi-Party Computation with Identifiable Abort in the dishonest majority setting with broadcast. SA stands for stand-alone, UC stands for Universal Composability [11], t is the max. number of corrupted parties, and $\beta := \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$. π_{OT} denotes any adaptively secure Oblivious Transfer (OT) protocol, $\mathcal{F}^n_{\mathsf{CRS}}$ is the Common Reference String (CRS) functionality from [12], $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ is the Correlated-Randomness functionality from [30]. $\mathcal{F}^n_{\mathsf{COM},1:1}$ is a one-to-one commitment and $\mathcal{F}^n_{\mathsf{SFE},f}$ is the Secure Function Evaluation functionality; both defined in Appendix B.3. Note that the impossibility 6) does not contradict 3) because 6) does not assume a CRS.

1.3 Technical Overview

We state our results in the standard UC-framework (see Appendix B.2) in terms ideal functionalities and protocols that realize them. Due to the large notational and conceptual overhead of rigorous statements about MPC and the given space limitations, we decided to give most of the formal definitions and statements in the appendices, and instead try to convey the core idea behind our techniques in this overview and how they are combined to obtain our main results.

The main idea behind the usage of the Trust Graph (TG) is that if the TG is disconnected, then there are at least two partitions A and B such that all parties in A distrust all parties in B. Now, if honest parties always trust each other (as is the case throughout this paper), then all honest parties must be in the same connected component. W.l.o.g. let all honest parties be in A, then they can jointly identify B and abort with $(\texttt{abort}, B)$. In this sense the disconnectivity of the TG is equivalent to the identification of malicious parties, and hence the abort of a ID-MPC protocol.

To simplify our analysis we formally introduce an ideal functionality $\mathcal{F}^n_{\mathsf{TG}}$ for n parties in Appendix C (see Definition 7). This functionality stores an (initially complete) graph with one vertex per party. Any party P can announce "conflicts" by sending $(\texttt{conflict}, \mathsf{P}')$ to $\mathcal{F}^n_{\mathsf{TG}}$. Consequently, the edge $(\mathsf{P}, \mathsf{P}')$ is irrevocably removed from the TG—we say P and P' are *in conflict*. Any party can also query the currently stored graph $G = (P, E)$ (typically at the onset of each round); such that all parties have a consistent view of the TG in each round. The functionality $\mathcal{F}^n_{\mathsf{TG}}$ can be viewed as syntactical sugar, as it can be perfectly securely realized using only broadcast (see Lemma 6).

Now, we give a high-level intuition of a particular set of protocol rules that will prove useful in our results for the correlation complexity. We call this set of six rules "abort-respecting" (see Definition 8). One can view it as a kind of manual for how to utilize the Trust Graph. Informally, abort-respecting protocols ensure in particular the following properties:

- Honest parties are never in conflict.
- Whenever some party has strictly more than t conflicts, it must be malicious.
- Whenever some setup of cardinality β is aborted,[6] the subTG on the participants of the setup becomes disconnected.
- When the protocol aborts (with $(\texttt{abort}, C)$ where C is a set of malicious parties), the overall TG is disconnected.
- When the overall TG becomes disconnected, the protocol aborts (with $(\texttt{abort}, C)$ where C is a set of malicious parties).

Intuitively, from these rules and the usage of $\mathcal{F}^n_{\mathsf{TG}}$ follows Lemma 1. More formally, it states that any secure protocol for some functionality in some F-hybrid model can be transformed into a secure and abort-respecting protocol for the same functionality in the $F \cup \{\mathcal{F}^n_{\mathsf{TG}}\}$-hybrid model (compare Lemma 7). As a corollary, we note that to rule out *all* ID-MPC protocol for some functionality it suffices to rule out all abort-respecting protocols. We will use this fact in the proof of our lower bound on the CC.

Before, we want to elaborate a bit on the third property. When a setup with Identifiable Abort is aborted, all participants P obtain the message $(\texttt{abort}, C)$ where $C \subseteq P$ is some set of malicious participants. Then all honest parties declare conflicts with C via the $\mathcal{F}^n_{\mathsf{TG}}$ functionality. In the next round either a) all parties $P \setminus C$ declared conflicts with the identified parties C, or b) there are some "loyalists" $L \subseteq P \setminus C$ who did not declare conflicts with *all* identified parties C. In the first case the subTG is clearly disconnected between $P \setminus C$ and C. In the second case b) note that loyalists noticeably deviate from the abort-respecting rules; thus the honest parties add the loyalists L to the identified parties C and repeat the procedure. Since in each iteration at least one loyalist gets added to the identified parties, the overall procedure terminates within at most n iterations.

Lower Bound on the Correlation Complexity Eventually, we show that no protocol in the $\{\mathcal{F}^n_{\mathsf{BC}}, \mathcal{F}^{\beta-1}\}$-hybrid model for any functionality $\mathcal{F}^{\beta-1}$ of cardinality $\beta - 1$ can securely realize an ideal commitment functionality where n is the overall number of parties, t is a upper bound on the malicious parties and $\beta := \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$. Towards this end, we prove a graph-theoretical lemma that relates the connectivity of the overall TG to the connectivity of its subgraphs. More concretely, the lemma constructs a connected graph that has "many" disconnected subgraphs of the cardinality $\beta - 1$. With the intuition that aborts of setups correspond to disconnected subTGs (via the abort-respecting property), this graph-theoretical lemma translates into a strategy for the adversary to abort many setups in a clever way such that the overall TG remains connected, i.e., the overall protocol cannot abort. However, after these many setups are aborted, we follow the idea of Canetti and Fischlin [12]

[6] Recall that we only assume setups to have security with Identifiable Abort.

to prove that any protocol that only relies on the remaining setups must either violate the hiding or the binding property. We note that this proof strategy only works because we want to rule out composable commitment protocols.
The high-level idea is as follows: Because "many" setups are aborted, the sender cannot "directly" commit towards the receiver via some setup that contains both the sender *and* the receiver. Consequently, in order to be committed towards the receiver (binding), the sender has to send the message (information-theoretically) to intermediate parties—even when all parties act honestly (relative to their view of the Trust Graph given the aborted setups) in the commitment phase. However, this set of intermediate parties is small enough that an alternative environment can corrupt it (because $t \geq n/2$) and thus extract the message of an honest sender during the commitment phase (not hiding).

Lemma 2 (Connected graph $\implies$ many disconnected $\beta-1$-subgraphs). *Let $n, t \in \mathbb{N}$ s.t. $n/2 \leq t \leq n-1$, and let $\beta \coloneqq \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$. Furthermore, let V be a set of n vertices and let $v, v' \in V : v \neq v'$ be two different vertices. There exist some edges $E \subseteq \binom{V}{\leq 2}$ s.t.*

1. *$G \coloneqq (V, E)$ is an undirected, reflexive and* connected *graph,*
2. *$\forall \{u, u'\} \in E : |\mathrm{N}_G(u) \cap \mathrm{N}_G(u')| \geq n - t$,*
3. *for each $V' \in M$ the subgraph $G' \coloneqq (V', E \cap \binom{V'}{\leq 2})$ is disconnected*

where

$$N_u \coloneqq \begin{cases} \{u\} & \text{, if } t = n-1 \\ \mathrm{N}_G(u) & \text{, else} \end{cases}$$

is the set of "effective" neighbors of any vertex u, and

$$M \coloneqq \{V' \subseteq V \mid \quad V' \cap N_v \neq \emptyset \quad \wedge \quad V' \cap N_{v'} \neq \emptyset \quad \wedge \quad |V'| < \beta\}$$

is the set of relevant subsets of vertices that contain both an effective neighbor of v and an effective neighbor of v'.

The proof is contained in Appendix D.1. For $t = n - 1$ the lemma states that there exists some graph whose subgraphs G' that contain both v and v' are disconnected, yet the overall graph is connected. For $t \leq n-2$ the lemma states that there exists some graph whose subgraphs G' that contain both a neighbor[7] of v and a neighbor of v' are disconnected while the overall graph is connected.

Application to ID-MPC. Throughout, we denote the security parameter by λ. In the context of our impossibility proof, the graph G takes the role of the Trust Graph, $v = \mathsf{S}$ will be the sender, and $v' = \mathsf{R}$ will be the receiver. As such, the lemma translates to the statement that all setups in which the sender and the receiver (or their neighbors respectively) participate jointly can be aborted by the adversary without causing the overall TG to become disconnected, thus evading identification. The proof is essentially just a constructive description of the graph G alongside a proof of its properties. This graph-theoretic statement translates into the context of ID-MPC protocols as follows:

[7] Note that is vertex is their own neighbor because the graph is reflexive.

Corollary 1. *Let $n = n(\lambda), t = t(\lambda), \beta = \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$ s.t. $0 \leq t < n$. For any security parameter $\lambda \in \mathbb{N}$ let P_λ be a set of n parties, and let $v_\lambda, v'_\lambda \in P_\lambda$ be two different parties. Furthermore, let π^F be any abort-respecting protocol for some functionality $\mathcal{F}^n$ in some F-model s.t. $\mathcal{F}^n_{\mathsf{BC}} \in F$. An adversary for π^F that corrupts t parties can abort all setups of cardinality at most $\beta - 1$ in which any effective neighbor of v_λ and any effective neighbor of v'_λ participate, without disconnecting the overall Trust Graph G.*

For $t \geq n/2$ this follows from Lemma 2 and Corollary 5 (aborted setups correspond to disconnected subgraphs). Also, for $t < n/2$ it follows that $\beta = 1$, hence Corollary 1 follows trivially. Finally, we get the formal statement.

Theorem 2 (No transmitted commitment). *Let $n = n(\lambda), t = t(\lambda), \beta :=\min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$ s.t. $n/2 \leq t < n$ and $\binom{n}{\beta} \in \mathrm{poly}(\lambda)$. No $\{\mathcal{F}^2, ..., \mathcal{F}^{\beta-1}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid protocol can securely UC-realize $\mathcal{F}^n_{\mathsf{COM},1:1}$ against environments that (maliciously) corrupt up to t parties. Formally, we get*

$$\left\{\mathcal{F}^2, ..., \mathcal{F}^{\beta-1}, \mathcal{F}^n_{\mathsf{BC}}\right\} \overset{\text{comp}}{\not\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{COM},1:1} \tag{1}$$

where $\mathcal{F}^2, ..., \mathcal{F}^{\beta-1}$ stand for arbitrary functionalities of the respective cardinality, and $\mathcal{F}^n_{\mathsf{COM},1:1}$ is defined in Appendix B.3.
Consequently, the correlation complexity for UC-secure ID-MPC is at least β.

The proof is contained in Appendix D.1.

Corollary 2. *Let $n = n(\lambda)$. In particular, we find $\{\mathcal{F}^2, ..., \mathcal{F}^{n-3}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{comp}}{\not\rightsquigarrow}_{n-2} \mathcal{F}^n_{\mathsf{COM},1:1}$ where $\mathcal{F}^2, ..., \mathcal{F}^{n-3}$ stand for arbitrary functionalities of the respective cardinality. This shows that the result $\{\mathcal{F}^{n-1}_{\mathsf{Corr},\mathcal{D}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\rightsquigarrow}_{n-2} \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ from [41] is tight up to a constant of 1.*[8]

Upper Bound on the Correlation Complexity Towards our construction, we first prove a complementary graph-theoretical lemma that relates the connectivity of the overall TG to the connectivity of its subgraphs. More concretely, the lemma states that any graph with "many" disconnected subgraphs of the cardinality β must be disconnected.

Lemma 3 (Connected graph $\implies$ few disconnected β-subgraphs). *Let $n, t \in \mathbb{N}$ s.t. $n/2 \leq t \leq n-2$, and let $\beta := \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$. Let V be a set of n vertices and let $v, v' \in V : v \neq v'$ be two different vertices. Moreover, let $E \subseteq \binom{V}{\leq 2}$ be a set of edges s.t. $G := (V, E)$ is an undirected, reflexive graph, and let $N_u := \mathrm{N}_G(u)$ be the set of neighbors of any vertex u, let*

$$M := \{V' \subseteq V \mid \quad V' \cap N_v \neq \emptyset \quad \wedge \quad V' \cap N_{v'} \neq \emptyset \quad \wedge \quad |V'| = \beta\}$$

be the set of relevant subsets of vertices that contain both a neighbor of v and a neighbor of v', and let

$$E^* := \{\{u, u'\} \in E \mid |N_u \cap N_{u'}| \geq n - t\}$$

[8] We note that [41] state their results in the stand-alone model.

be the set of postprocessed[9] *edges. If for all $V' \in M$ the subgraph $G' := (V', E \cap \binom{V'}{\leq 2})$ is disconnected, then $G^* := (V, E^*)$ is disconnected. Furthermore, the map $\phi : G \mapsto G^*$ is efficiently computable.*

This lemma tightly complements Lemma 2. It states that as soon as all subgraphs G' that contain both a neighbor of v and a neighbor of v' are disconnected, the overall postprocessed graph G^* must be disconnected as well. The proof is contained in Appendix D.2.

Application to ID-MPC. In the context of our construction, the graph G takes the role of the TG, $v = \mathsf{S}$ will be the sender, and $v' = \mathsf{R}$ will be some receiver. As such, the lemma translates to the statement that at least one (not necessarily fixed) setup in which the sender and the receiver (or their neighbors respectively) participate jointly *cannot* be aborted by the adversary without causing the overall TG to become disconnected. This "guaranteed" setup can then reliably perform the commitment (resp. OT) between the sender and the receiver (resp. their neighbors).

The proof is by contradiction. Suppose all subgraphs G' are disconnected, yet G^* were connected. Then there must be a path W from any neighbor $u \in N_v$ to any neighbor $u' \in N_{v'}$ with length $\Delta_{G^*}(u, u') > \beta$. Note that, by definition of E^*, all adjacent parties in G^* must have at least $n - t$ common neighbors. This means that the parties along the path W must have many auxiliary neighbors. Counting the overall number of parties yields a contradiction. We formalize this in Appendix D.2. For a visual representation see Fig. 1, note that any shortest path from N_1 to N_n always has length β (in the number of vertices). This graph-theoretic statement translates into the context of ID-MPC protocols as follows:

Corollary 3. *Let $n = n(\lambda), t = t(\lambda), \beta := \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$ s.t. $n/2 \leq t \leq n-2$. For any security parameter $\lambda \in \mathbb{N}$ let P_λ be a set of n parties, and let $v_\lambda, v'_\lambda \in P_\lambda$ be two different parties. Furthermore, let π^F be any abort-respecting protocol for some functionality $\mathcal{F}^n$ in some F-model s.t. $\mathcal{F}^n_{\mathsf{BC}} \in F$. If an adversary for π^F that corrupts at most t parties aborts all setups of cardinality β in which any neighbor of v_λ and any neighbor of v'_λ participate, then the overall Trust Graph becomes disconnected, i.e., the protocol π^F aborts.*

This follows from Lemma 3 and Corollary 5 (aborted setups correspond to disconnected subgraphs).

In particular, in our protocols we require all honest parties to locally postprocess $G^* = \phi(G)$ from Lemma 3 when querying the TG G from $\mathcal{F}^n_{\mathsf{TG}}$. Moreover, we require all parties to abort according to G^* instead of G. This modification of the abort condition is justified because the additional (specific) conflicts introduced by the postprocessing ϕ preserve the invariant that no two honest parties are in conflict, as required by Rule 6 of Definition 8.

[9] The postprocessing ϕ corresponds to the (repeated) application of Rule 4 of Definition 8, i.e., removing edges from parties with strictly more than t conflicts.

Committed Oblivious Transfer. Before we proceed with a more detailed description of our protocols we have to introduce a committed variant of Oblivious Transfer (OT) [20, 22] where the sender, the receiver and some witnesses participate. We call this variant Fully Committed Oblivious Transfer (FCOT). As in the standard 1-out-of-2 OT, the sender inputs two messages and the receiver inputs a choice bit, then the receiver obtains its chosen message while the receiver remains oblivious to the choice bit. The committed variant additionally allows the sender and the receiver to later open their inputs to all other parties (called witnesses).

The purpose of this FCOT can be state as follows.

Lemma 4 (Completeness of committed OT (informal)). *There is a protocol in the $\mathcal{F}^n_{\mathsf{FCOT}}$-hybrid model that realizes any ideal n-party functionality.*

We can replace the standard OT setups in the IPS-compiler [31]. The IPS-compiler is an OT-hybrid protocol in the client-server-model that realizes general MPC guaranteeing security with (non-identifiable) abort against malicious (active) adversaries. In this protocol each party sets up a watchlist for each server such that other parties can monitor a small subsets of servers to detect tampering with overwhelming probability. Once a party detects misbehavior on some server it announces a complaint and all parties abort the protocol (without identifying malicious parties). For this reason, the standard IPS-compiler only enjoys security with (non-identifiable) abort. Substituting all calls to classical OT setups with calls to FCOT setups allows the parties to open all messages regarding the server in question. This way all parties can retrace which party misbehaved, thus identifying at least one malicious party, hence the resulting protocol enjoys security with Identifiable Abort. (We refer the interested reader to Appendix D.3 for more details.)

We continue with a high-level overview of our two protocols that utilize the guaranteed setups mentioned above. The two constructions are

- n-party commitment from β-party commitments and n-party broadcast, and
- n-party FCOT from β-party FCOT, n-party commitments and broadcast.

Commitment expansion.

Theorem 3 (COM expansion). *Let $n = n(\lambda), t = t(\lambda), \beta := \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$ s.t. $n/2 \leq t \leq n-2$ and $\binom{n}{\beta} \in \mathrm{poly}(\lambda)$. There is an efficient protocol π_{COM} that statistically securely UC-realizes $\mathcal{F}^n_{\mathsf{COM}}$ in the $\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against environments that (maliciously) corrupt up to t parties. Formally,*

$$\left\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta, \mathcal{F}^n_{\mathsf{BC}}\right\} \overset{\text{stat}}{\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{COM}} \ . \tag{2}$$

On a high level our one-to-many commitment protocol follows a commit-and-prove approach. Without going into too much detail, we outline the idea of the protocol. The sender inputs its message m—in the form of a threshold sharing μ—into all setups[10] and gives (secret-shared) masks ξ^j to its neighbors

[10] The sender inputs the same shares into each setup that it participates in.

$\mathsf{R}_j \in \mathrm{N}_G(\mathsf{S})$ who, in turn, also commit to their sharings in all commitment setups. Additionally, the sender broadcasts the message's sharing μ masked with the masks' sharings $\sigma \coloneqq \mu \oplus \bigoplus_{\mathsf{R}_j \in \mathrm{N}(\mathsf{S})} \xi^j$. Subsequently, all parties broadcast some randomly drawn "probing indices" on which the sender (resp. neighbors) broadcast the resp. share and open the setup commitment for the resp. share. Then all parties check for inconsistencies. Indeed, all setups (intended for the same value) contain sharings of the *same* (possibly masked) value with overwhelming probability. If shares differ significantly, this discrepancy will be detected with overwhelming probability; then the affected setup is considered aborted by identifying the committer as malicious. If shares differ only on a few indices, then the sharing's error-detection will allow the parties to notice that the shares are invalid. Again, the affected setup is considered aborted by identifying the committer as malicious. Moreover, due to the privacy of the secret-sharing opening a few shares does not reveal anything about the encoded value.
Lemma 3 guarantees that at least one setup of cardinality β that contains both a neighbor of the sender and a neighbor of the receiver must succeed. Otherwise, if all such setups are aborted, then the TG becomes disconnected by Lemma 3 and the honest parties can abort the protocol.

To open the message, all parties open all commitment setups and at least one honest receiver is able to recover the message either directly from the sender's sharing μ of the message, or from the opened masks ξ^j and the previously broadcasted masked sharing σ. Those receivers then broadcast the recovered message. Any honest receiver that did not receive any opening information—because all its setups have been aborted—then it outputs the majority of its neighbors' broadcasted messages. To see why such "cut-off" receivers output the correct message we have to see the following fact. Whenever all setups containing both a neighbor of the sender *and* a neighbor of the receiver are aborted, then an honest receiver has a majority of honest neighbors that could reconstruct the message. This statement follows from graph-theoretical considerations best visualized in Fig. 1. An intuitive explanation is that if all such setups are aborted, then the sender and the cut-off receiver have a large distance of at least β in the TG. In that case the cut-off receiver cannot have too many malicious neighbors, yet honest parties always remain neighbors.

Skipping ahead to the proof of security, the simulator will extract the committed message from the sharings input into the partial commitment setups. As the verification step ensures consistency among the committed messages the simulator's extracted message is uniquely defined and correct.

Committed OT expansion.

Theorem 4 (FCOT expansion). *Let $n = n(\lambda), t = t(\lambda), \beta \coloneqq \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$ s.t. $n/2 \le t \le n-2$ and $\binom{n}{\beta} \in \mathrm{poly}(\lambda)$. There is an efficient protocol π_{FCOT} that statistically securely UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta_{\mathsf{SFE},f}, \mathcal{F}^n_{\mathsf{COM}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against environments that (maliciously)* *corrupt up to t parties. Formally, for some specific functionality $\mathcal{F}^{\beta}$ we get*

$$\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^{\beta}, \mathcal{F}^n_{\mathsf{COM}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{FCOT}} \ . \tag{3}$$

Recall that in the FCOT functionality there exists a sender, a receiver and $n-2$ witnesses. Let $\mathcal{F}^{\beta}_{\mathsf{SFE},f}$ be some Secure Function Evaluation (SFE) setup for β-parties for some function f_{OT} that allows for an FCOT but whose details we omit at this point. In our FCOT protocol the sender and the receiver try to perform the global FCOT *directly*, i.e., via some setup in which both the sender and the receiver and $n-\beta$ witnesses (w.r.t. the overall FCOT) are left out. To ensure consistency with the excluded witnesses the sender and the receiver globally commit to their inputs (again as secret-sharings) via $\mathcal{F}^n_{\mathsf{COM}}$. Accordingly, the setup $\mathcal{F}^{\beta}_{\mathsf{SFE},f}$ also takes the same sharing as inputs. As in the commitment protocol the sender and the receiver open their shares, in the $\mathcal{F}^{\beta}_{\mathsf{SFE},f}$ setup and the global commitments, on some random probing indices to detect inconsistencies. To open, the sender and the receiver can simply open the global commitments to their respective inputs.

In the security proof the simulator extracts the sender's messages and the receiver's choice bit from their inputs to their global commitments. Again, the verification step (commit-then-prove by probing random shares) guarantees that the simulator's extracted inputs match the ones output be the honest parties in the real protocol execution with overwhelming probability.

As in the previous construction, we leverage Lemma 3 which guarantees essentially that some such "direct" SFE setup must not be aborted, if the protocol is not to abort. Here lies the technical difficulty of our protocol because Lemma 3 only guarantees such a setup between a neighbor of the sender and a neighbor of the receiver (not the sender and the receiver themselves). To remedy this issue we make the following observation: In the seemingly hopeless scenario where the adversary chooses to abort the setups in such a way that the sender and the receiver themselves are not able to perform the direct setup, then one of them (the honest one) has many honest neighbors. In this case the sender and receiver use their neighbors respectively to carry out the OT for them. Here, the sender and the receiver secret-share their inputs to retain their privacy and distribute them to their neighbors. While for the sender's messages the sharing seems straightforward (additive sharing), it may not be obvious how the receiver's choice bit can be shared s.t. the receiver obtains its chosen message. To this end we invoke a technique akin to the one used by Wolf and Wullschleger [43] which they used to show the symmetry of OT. This allows the receiver to only distribute additive shares of its choice bit (ensuring privacy) but still obtain the chosen message.

Equivalence of SFE-Complete Setups In Appendix D.3 we prove that the setups Fully Committed Oblivious Transfer (FCOT) $\mathcal{F}^n_{\mathsf{FCOT}}$, Secure Function Evaluation (SFE) $\mathcal{F}^n_{\mathsf{SFE},f}$, and Correlated-Randomness from [30] $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ can be efficiently realized from each other with statistical security; so we can substitute one with the other by the Universal Composability Theorem of [11]. (We include formal definitions of these functionalities in Appendix B.3.)

Putting the Results Together For brevity we use the short notation $F \rightsquigarrow_t \mathcal{F}$ (see Notation 8) to describe the construction of UC-secure protocols for the ideal functionality $\mathcal{F}$ in the F-hybrid model against at most t corruptions.

Corollary 4 (Composition of constructions). *The correlation complexity for UC-secure ID-MPC is at most β. We observe*

$$\left\{\mathcal{F}^{\beta}_{\mathsf{FCOT}}\right\} \overset{\text{stat}}{\rightsquigarrow}_{\beta} \mathcal{F}^{\beta}_{\mathsf{SFE},f} \tag{4}$$

$$\left\{\mathcal{F}^{2}_{\mathsf{SMT}}, \mathcal{F}^{\beta}_{\mathsf{SFE},f}, \mathcal{F}^{n}_{\mathsf{BC}}\right\} \overset{\text{stat}}{\rightsquigarrow}_{t} \mathcal{F}^{n}_{\mathsf{FCOT}} \tag{5}$$

$$\overset{\text{stat}}{\rightsquigarrow}_{n} \mathcal{F}^{n} \tag{6}$$

where $\mathcal{F}^n$ is any arbitrary functionality. Equation (5) *follows from the combination of Theorems 3 and 4, and Eqs.* (4) *and* (6) *follows from the ID-MPC-completeness of FCOT (see Lemma 8). For statistical security we get*

$$\left\{\mathcal{F}^{\beta}_{\mathsf{Corr},\mathcal{D}}\right\} \overset{\text{stat}}{\rightsquigarrow}_{\beta} \mathcal{F}^{\beta}_{\mathsf{FCOT}} \tag{7}$$

$$\Longrightarrow \left\{\mathcal{F}^{2}_{\mathsf{SMT}}, \mathcal{F}^{\beta}_{\mathsf{Corr},\mathcal{D}}, \mathcal{F}^{n}_{\mathsf{BC}}\right\} \overset{\text{stat}}{\rightsquigarrow}_{t} \mathcal{F}^{n} \tag{8}$$

where $\mathcal{F}^n$ is any arbitrary functionality. Equation (7) *follows from the ID-MPC-completeness of the correlated randomness setup (Theorem 6 in [30]). For computational security we get*

$$\left\{\pi_{\mathsf{OT}}, \mathcal{F}^{\beta}_{\mathsf{CRS}}, \mathcal{F}^{\beta}_{\mathsf{BC}}\right\} \overset{\text{comp}}{\rightsquigarrow}_{\beta} \mathcal{F}^{\beta}_{\mathsf{FCOT}} \tag{9}$$

$$\Longrightarrow \left\{\pi_{\mathsf{OT}}, \mathcal{F}^{2}_{\mathsf{SMT}}, \mathcal{F}^{\beta}_{\mathsf{CRS}}, \mathcal{F}^{n}_{\mathsf{BC}}\right\} \overset{\text{comp}}{\rightsquigarrow}_{t} \mathcal{F}^{n} \tag{10}$$

where $\mathcal{F}^n$ is any arbitrary functionality, π_{OT} is any adaptively secure OT protocol and $\mathcal{F}^{\beta}_{\mathsf{CRS}}$ is the Common Reference String functionality from [12]. Equation (9) *follows from the computational construction in Theorem 12 in [30].*

For ID-MPC with statistical security this reduces the required correlation complexity from n to β. For ID-MPC with computational security this reduces the required cardinality of the CRS for the computationally secure offline phase of the construction in [30] from n to β.

Theorem 1 (Correlation complexity bounds). *The correlation complexity for UC-secure Multi-Party Computation with Identifiable Abort (given broadcast) is $\beta := \min(n, \lfloor n/(n-t)\rfloor + \lceil n/(n-t)\rceil - 2) \approx 2n/(n-t)$ where n is the overall number of parties and t is an upper bound on the number of malicious parties. In other words, for any n-party functionality with Identifiable Abort there exists a protocol that uses hybrid functionalities of cardinality β and broadcast, but there exists an n-party functionality with Identifiable Abort which cannot be realized by any protocol that uses hybrid functionalities of cardinality $\beta - 1$ and broadcast.*

Proof. The theorem follows directly from Theorem 2 and Corollary 4. □

Supplementary Material

A Discussion

A note on efficiency. We view this work as taking the first step towards constructing protocols that resist a dishonest majority with small correlation complexity. As such we did not optimize the concrete efficiency of our protocols, to keep them as easily accessible as possible. Though, in certain settings the efficiency considerations in Section 1 still attest a (potentially exponential) advantage of our protocol for $t \leq (1-\varepsilon)n$, over protocols that use an n-party setup. Let us make some high-level remarks. First, our constructions only use information-theoretic tools, e.g. secret-sharing and some efficient graph operations. As such our protocol is computationally inexpensive, in particular it does *not* require any computational assumptions.

On the other hand, the communication complexity is comparatively high because the consistency of the Trust Graph is ensured by each party broadcasting their conflicts. The abort-respecting property which is at the heart of our technique requires that—upon abort of a setup—parties gradually declare conflicts until the corresponding subgraph is disconnected. This gradual declaration of conflicts (abort-then-disconnect) requires $\mathcal{O}(\beta) = \mathcal{O}(n/(n-t))$ rounds in the worst-case (where β is the cardinality of the setups). Moreover, the adversary can abort $\mathcal{O}(n^2)$ setups. This abort-then-disconnect approach leads to a worst-case round complexity of $\mathcal{O}(n^3/(n-t))$. This problem (with a similar bound $\mathcal{O}(n^2)$) also seems to arise in the concurrent work [41] and is thus not specific to our approach. Since it is not clear if this is an inherent problem when constructing MPC from smaller setups or not, we propose this question for further research.
At first glance replacing *one* large setup with *many* smaller setups may seem like a unfavorable tradeoff. However, all these smaller setups can be performed *concurrently* in the same rounds because they are independent of each other. Thus, while the communication complexity might increase proportional to the number of setups (polynomial in λ), the number of rounds only incurs an *additive overhead*.
Overall, in our protocols the computational complexity is low, the communication complexity is high, and the round complexity is quadratic[11] in n for $t \leq (1-\varepsilon)n$.

On the use of broadcast. To us, it not known if a *full* broadcast is indeed necessary for all $t \geq n/2$. Cohen and Lindell [17] show that without full broadcast some functionalities are not realizable for $t \geq n/3$ (see Corollary 1.5 in [17]). To some degree this justifies that our technique relies on full n-wise broadcast. Although we see potential to generalize our technique to partial broadcasts for fewer corruptions; however, we leave this question for future work. Nevertheless,

[11] This is actually the additive overhead on top of the protocol of Ishai, Ostrovsky, and Zikas [30].

in light of our lower bound, we can state that partial broadcasts must have cardinality at least β.

Limitation of the overall number of parties. Our protocols need $\binom{n}{\beta} \in \mathrm{poly}(\lambda)$ to have polynomial runtime. The reason for this is that there are $\binom{n}{\beta}$ subsets of parties of size β almost all of which might have to be used, if the adversary aborts the setups in a careful way. Unfortunately, this limitation seems inherent in all protocols that don't discriminate between different subsets of parties in any meaningful way. In other words, to overcome this limitation a protocol would have to *never* use setups on some a priori fixed subset of parties of size β although they are not aborted. This gives the adversary more leeway to abort the other setups in a way that breaks the protocol.

The construction of [41] only supports arbitrary $n \in \mathrm{poly}(\lambda)$ for constant expansions $n - s \in \mathcal{O}(1)$ (where s is the setup size) because of the exponential composition blowup of the runtime. We have a different situation; see Table 1 for an overview of the supported number of parties vs. honest parties. The first intuitive thing to note is that the smaller the fraction of honest parties the less overall parties are supported. The case $t = n-1$ is trivial. Somewhat counterintuitive is the case for $t = n - 2$ which supports polynomially many parties, the reason for this is that for $t = n - 2$ the necessary and sufficient setup encompasses all parties but two, i.e. $\beta = n-2$. Here all but one setup contains at least one honest party.
Another case worth mentioning is the two-party case $n = 2$ and $t = n - 1$ where security with abort is trivially equivalent to IA; interestingly $\beta = 2$ carries over to larger $n = 2t \geq 4$ from the honest majority case.
When an arbitrarily small but constant fraction of parties is honest our protocols are efficient for any $n \in \mathrm{poly}(\lambda)$. However, when less than a constant fraction of parties is honest, e.g. $\Theta(n/\ln n)$, then the overall number parties drops drastically below $n \in \mathcal{O}(\exp\sqrt{\ln\lambda}) \subset \lambda^{o(1)}$. To relativize, these bounds only apply when trying to design protocols with IA from *minimal* setups. It could be the case that slightly larger setups yield protocols that support many more overall parties.

B Technical Preliminaries

B.1 Definitions & Notation

We use λ for the (statistical) security parameter, n for the overall number of parties, h for the lower bound on the number of honest parties, and t for the upper bound on number of malicious parties. We also use $\mathrm{negl}(\lambda)$ and $\mathrm{owhl}(\lambda)$ to denote the set of negligible resp. overwhelming functions w.r.t. λ.

Notation 1 (Order of functions). *For any two functions $f, g : \mathbb{N} \to \mathbb{R}$ we write $f \leq g \iff \forall \lambda \in \mathbb{N} : f(\lambda) \leq g(\lambda)$.*

Notation 2 (Subsets). *For any set V and $k \in \mathbb{N}$ we denote the set of subsets of cardinality k by $\binom{V}{k} \coloneqq \{V' \subseteq V \mid |V'| = k\}$. Also, we use $\binom{V}{\leq k} \coloneqq \bigcup_{\kappa=1}^{k} \binom{V}{\kappa}$.*

Notation 3 (Union of disjoint sets). *For the reader's convenience use a special notation for the union of two* disjoint *sets. Whenever we write $V \uplus V'$ instead of $V \cup V'$ for any two sets V and V' it holds that $V \cap V' = \emptyset$. In particular, we use the fact that $|V \uplus V'| = |V| + |V'|$. To avoid confusion, note that $V \uplus V$ does* ***not*** *mean the exclusive union $(V \cup V') \setminus (V \cap V')$.*

Notation 4 (Neighbors). *For an undirected graph $G = (V, E)$ we use the following notation for neighbors: $\mathrm{N}_G(v) \coloneqq \{v' \mid \{v, v'\} \in E\}$.*

Notation 5 (Distance). *For any undirected graph $G = (V, E)$ and any two vertices $v, v' \in V$ we denote by $\Delta_G(v, v')$ the number of vertices on the shortest path between v and v' (including v and v').*

Notation 6 (Index sets). *For any $k_0, k_1 \in \mathbb{N} : k_1 \geq k_0$ we write $[k_0] \coloneqq \{1, ..., k_0\}$ and $[k_0, k_1] \coloneqq \{k_0, k_0 + 1, ..., k_1 - 1, k_1\}$.*

Notation 7 (Lists). *For any list $\gamma \coloneqq (\gamma_i)_{i \in [l]}$ of length $l \in \mathbb{N}$ we denote the (filtered) list at index set $\nu \subseteq [l]$ by $\gamma_\nu \coloneqq (\gamma_i)_{i \in \nu}$.*

Lemma 5 (Error detection). *Let $u, w, s \in \mathbb{N}_+$ s.t. $s, w \leq u$ and let $W \in \binom{[u]}{w}$, then*

$$\mathrm{Pr}_{S \leftarrow \binom{[u]}{s}}[W \cap S = \emptyset] = \prod_{i=0}^{s-1} \frac{u - w - i}{u - i} \leq \prod_{i=0}^{s-1} \frac{u - w}{u} = (1 - w/u)^s \leq 2^{-sw/u} \ . \tag{11}$$

Definition 1 (Threshold secret sharing). *For any $\ell_1 = \ell_1(\lambda)$ and $\ell_2 = \ell_2(\lambda)$ s.t. $\ell_1 \leq \ell_2$ an (ℓ_1, ℓ_2)-threshold secret sharing scheme for message space $M = (M_\lambda)_{\lambda \in \mathbb{N}}$ is defined by a probabilistic algorithm $\mathrm{Share}_{\ell_1,\ell_2}$ and a deterministic algorithm $\mathrm{Recover}_{\ell_1,\ell_2}$ that compute the following (family of) functions:*

- $\mathrm{Share}_{\ell_1,\ell_2} : M_\lambda \to (\{0,1\}^\lambda)^{\ell_2} : m \mapsto \mu = (\mu_\kappa)_{\kappa \in [\ell_2]}$
  $\mathrm{Share}_{\ell_1,\ell_2}$ *takes a message $m \in M_\lambda$ and outputs ℓ_2 shares such that* $\mathrm{Recover}_{\ell_1,\ell_2}$ *reconstruct the message but $\ell_1 - 1$ shares perfectly hide the secret. Formally, for all possible shares $\mu' \in (\{0,1\}^\lambda)^b$ output by* $\mathrm{Share}_{\ell_1,\ell_2}$*, for all messages $m, m' \in M_\lambda$ and for all sets of indices $\nu \in \binom{[\ell_2]}{\ell_1 - 1}$ it should hold that*

$$\left|\mathrm{Pr}_{\mu \leftarrow \mathrm{Share}_{\ell_1,\ell_2}(m)}[\mu_\nu = \mu'_\nu] - \mathrm{Pr}_{\mu \leftarrow \mathrm{Share}_{\ell_1,\ell_2}(m')}[\mu_\nu = \mu'_\nu]\right| \in \mathrm{negl}(\lambda) \ .$$

- $\mathrm{Recover}_{\ell_1,\ell_2} : (\{0,1\}^\lambda)^{\ell_2} \to M_\lambda \uplus \{\bot\} : \mu \mapsto m$
  We require error-detection. Formally, for all messages $m \in M_\lambda$ and for all sets of indices $\nu \in \binom{[\ell_2]}{\ell_1}$ it should hold that

$$\mathrm{Pr}_{\mu \leftarrow \mathrm{Share}_{\ell_1,\ell_2}(m)}\begin{bmatrix} \forall \mu' \text{ s.t.} & : & \mu = \mu' \iff \mathrm{Recover}_{\ell_1,\ell_2}(\mu') = m \\ \mu_\nu = \mu'_\nu & & \mu \neq \mu' \iff \mathrm{Recover}_{\ell_1,\ell_2}(\mu') = \bot \end{bmatrix} \in \mathrm{owhl}(\lambda) \ .$$

For example, we could use Shamir's secret sharing [40] with $\ell_2 = 2\ell_1$. The purpose of these threshold sharing is to ensure that parties input the same shares across multiple setups.[12]

[12] The idea is conceptually similar to *public verifiability* [1].

Definition 2 (Realizing ideal functionalities). *Let $n = n(\lambda)$ and $t = t(\lambda)$ s.t. $t \leq n$. Let F be a set of setups (ideal functionalities) with cardinality at most n, let π^F be any family of protocols in the F-hybrid model, and let $\mathcal{F}^n$ be an ideal functionality. Iff π^F securely UC-realizes $\mathcal{F}^n$ against environments that (maliciously) corrupt at most t parties, we write*

$$\pi^F \geq_{\mathsf{t}} \mathcal{F}^n \ , \qquad \textit{or equivalently}$$

$$\exists \mathcal{S}\, \forall \mathcal{Z}_{\leq t} : \left\{\mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}\left(\mathcal{Z}_{\leq t(\lambda)}(1^\lambda, \cdot)\right)\right\}_{\lambda \in \mathbb{N}} \approx \left\{\mathsf{IDEAL}_{\mathcal{F}^{n(\lambda)}, \mathcal{S}}\left(\mathcal{Z}_{\leq t(\lambda)}(1^\lambda, \cdot)\right)\right\}_{\lambda \in \mathbb{N}}$$

where $\mathcal{Z}_{\leq t}$ denotes any environment that maliciously corrupts at most t parties, $\mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}(\mathcal{Z}_{\leq t(\lambda)}(1^\lambda, \cdot))$ is the environment's output when running with the (hybrid) protocol π^F on security parameter λ, and $\mathsf{IDEAL}_{\mathcal{F}^{n(\lambda)}, \mathcal{S}}(\mathcal{Z}_{\leq t(\lambda)}(1^\lambda, \cdot))$ is the environment's output when running with the simulator on security parameter λ. Here, $\mathcal{A}_\mathcal{D}$ is the canonical dummy adversary who simply forwards communication from and to the environment (see Section 4.3.1 in [10]).

Notation 8 (Protocol construction). *Let $n = n(\lambda)$ and $t = t(\lambda)$ s.t. $t \leq n$. For any set of setups F with cardinality at most n, and any ideal functionality $\mathcal{F}^n$, we write $F \rightsquigarrow_t \mathcal{F}^n$, iff there is a protocol π^F that securely UC-realizes $\mathcal{F}^n$ in the F-hybrid model against environments that (maliciously) corrupt up to t parties. More formally:*

$$F \rightsquigarrow_t \mathcal{F}^n \iff \exists \pi^F : \pi^F \geq_{\mathsf{t}} \mathcal{F}^n \ . \tag{12}$$

Conversely, we write

$$F \not\rightsquigarrow_t \mathcal{F}^n \iff \forall \pi^F : \pi^F \not\geq_{\mathsf{t}} \mathcal{F}^n \ . \tag{13}$$

We furthermore use the additional notation $F \overset{\text{stat}}{\rightsquigarrow}_t \mathcal{F}^n$ resp. $F \overset{\text{comp}}{\rightsquigarrow}_t \mathcal{F}^n$ to denote the construction is secure against a computationally unbounded resp. efficient environment.

Next, we recall the definition of minimal complete functionalities (setups) as introduced by Fitzi et al. [23]. The notion of a minimal complete cardinality (MCC) is relatively straightforward when considering any fixed number of parties $n \in \mathbb{N}_{\geq 2}$, i.e., when the number of parties does not depend on the security parameter. Then, as in the original work [23], for each $n, t \in \mathbb{N}_{\geq 2} : t \leq n$ there exists exactly one number $\kappa_t \in \{2, ..., n\}$ s.t. κ_t is the MCC. As such their definition gives a meaningful setup size for *each* $n \in \mathbb{N}$, i.e., it is not only an asymptotic statement.

Definition 3 (Concrete minimal complete cardinality [23]). *Let $n \in \mathbb{N}$ be any number of parties and $t \in \mathbb{N}$ be any corruption threshold s.t. $t \leq n$, and let $\mathcal{F}^n$ be an ideal functionality. Furthermore, let F be a set of setups for at most n parties.*
Relative to F, we say the cardinality $\kappa_{t, \mathcal{F}^n, F} \in \mathbb{N}$ is (concretely)

- complete, *iff $\exists \mathcal{F}^{\kappa_{t, \mathcal{F}^n, F}} : \{\mathcal{F}^{\kappa_{t, \mathcal{F}^n, F}}\} \cup F \rightsquigarrow_t \mathcal{F}^n$, and*
- minimal, *iff $\forall \kappa' < \kappa_{t, \mathcal{F}^n, F} : (\forall \mathcal{F}^{\kappa'} : \{\mathcal{F}^{\kappa'}\} \cup F \not\rightsquigarrow_t \mathcal{F}^n)$, i.e., if any setups of any cardinality κ' are insufficient for constructing $\mathcal{F}^n$.*

However, if the number of parties $n = n(\lambda)$ scales with the security parameter a more nuanced definition is necessary.

Definition 4 (Asymptotic minimal complete cardinality (MCC)). *Let $n = n(\lambda)$ be any number of parties and $t = t(\lambda)$ be any corruption threshold s.t. $t \leq n$, and let $\mathcal{F}^n$ be an ideal functionality. Furthermore, let F be a set of setups for at most n parties.*
Relative to F, we say the cardinality $k_{t,\mathcal{F}^n,F} = k_{t,\mathcal{F}^n,F}(\lambda)$ is (asymptotically)

- complete*, iff $\exists \mathcal{F}^{k_{t,\mathcal{F}^n,F}} : \{\mathcal{F}^{k_{t,\mathcal{F}^n,F}}\} \cup F \rightsquigarrow_t \mathcal{F}^n$, and*
- minimal*, iff*

$$\forall k' = k'(\lambda) : \left(\limsup_{\lambda\to\infty} k_{t,\mathcal{F}^n,F}(\lambda) - k'(\lambda) > 0 \implies \forall \mathcal{F}^{k'} : \left\{ \mathcal{F}^{k'} \right\} \cup F \not\rightsquigarrow_t \mathcal{F}^n \right) , \tag{14}$$

i.e., if any setups of any cardinality k' (that is less than $k_{t,\mathcal{F}^n,F}$ for infinitely many[13] λ) are insufficient for constructing $\mathcal{F}^n$.

Remark 1. Note that, technically, the asymptotic MCC makes no statement about whether setups of a certain *fixed* cardinality are sufficient to realize a certain functionality for a *fixed* number of parties. Instead, it only makes an asymptotic statement. The intuitive reason is that protocol runs with small security parameter λ (resp. number of parties n) are irrelevant because the indistinguishability of the real and the ideal execution is only an asymptotic property.[14] To make this point more clear, note that if $k_{t,\mathcal{F}^n,F}$ is an MCC, then so is

$$k' : \lambda \mapsto \begin{cases} k_{t,\mathcal{F}^n,F}(\lambda) & \lambda \geq \lambda' \\ 2 & \lambda < \lambda' \end{cases} \tag{15}$$

for any $\lambda' \in \mathbb{N}$. This demonstrates that the asymptotic MCC is not unique. As such, it only yields an asymptotic statement.
Though, both definitions coincide for fixed $n \in \mathbb{N}$, then we find that the concrete MCC of [23] is equal to $\kappa_{\tau,\mathcal{F}^n,F} = \limsup_{\lambda\to\infty} k_{t,\mathcal{F}^n,F}(\lambda)$ where $k_{t,\mathcal{F}^n,F}$ is any asymptotic MCC and $\tau = \limsup_{\lambda\to\infty} t(\lambda) \in \mathbb{N}$.

We stress that our results apply both to the case of a fixed $n \in \mathbb{N}_{\geq 2}$ as well as a variable $n = n(\lambda)$. In the former case the condition $\binom{n}{\beta} \in \text{poly}(\lambda)$ (necessary for our construction) holds trivially. Lastly, we want to mention that [23] defined the MCC relative to $F = \emptyset$ while in this work we consider $F = \{\mathcal{F}^n_{\mathsf{BC}}\}$, i.e., the broadcast functionality defined in Appendix B.3.

Definition 5 (Correlation complexity (CC)). *Let $n = n(\lambda)$ be any number of parties and $t = t(\lambda)$ be any corruption threshold s.t. $t \leq n$, and let F be a set of setups for at most n parties. We say $\beta(\lambda) := \max_{\mathcal{F}^n} k_{t,\mathcal{F}^n,F}(\lambda)$ is the correlation complexity relative to F and t where $\mathcal{F}^n$ is any ideal functionality for n parties. (In this work we only consider $F = \{\mathcal{F}^n_{\mathsf{BC}}\}$, i.e., the broadcast functionality defined in Appendix B.3, and we omit the argument λ.)*

[13] One can alternatively also define an "all-but-finite"-variant.

[14] To the best of our knowledge this issue has not been stated explicitly in previous literature.

B.2 Setting

Our constructions enjoy information-theoretic or **statistical security**, no computational assumptions are made. We only assume the existence of (ideal) hybrid functionalities—which we call *setups* to distinguish them semantically from the (to-be-constructed) ideal functionality. This leaves the means of the realization of these setups up to the user, e.g. via physical means such as trusted hardware [27, 39] or noisy channels [21, 19]. All functionalities are assumed to be *authenticated*. We focus on **static corruptions** of an arbitrary number of parties at the onset of the protocol. We denote the total number of parties by n and the upper bound for malicious parties by $t \leq n$.

Technically speaking, our results don't explicitly assume additional pairwise secure channels, since they can be emulated by setups of size at least 2. Adding pairwise channels does not affect our results, in fact we use pairwise channels as a conceptual simplification in our protocols. However, in our settings all parties have access to an n-party *broadcast*, which we model as ideal functionality $\mathcal{F}^{n}_{\mathsf{BC}}$. (See a brief discussion on the broadcast in Appendix A.)

The UC framework. We perform our analysis in the Universal Composability (UC) framework [10, 11], which is a strong version of simulation-based security [26, 25]. The key idea there is to compare a real protocol execution between mutually distrustful parties to an idealized execution, where a trusted party performs the computation based on the participants inputs. The behavior of the trusted party is specified by an *ideal functionality* $\mathcal{F}$. In the real world, all parties execute a protocol π, which is said to *realize* the functionality $\mathcal{F}$, if it can be shown to be indistinguishable from the ideal world. This requires a *simulator* who creates a transcript of an execution without knowing the parties' inputs. More precisely, the transcripts of both worlds must be indistinguishable for any non-participant, even those who know the parties' secret inputs. The transcript includes the output of all parties and the respective adversary. Indistinguishability of the two worlds implies that the real adversary cannot learn anything from the real protocol execution that the simulator cannot contrive without knowing the private inputs.

As opposed to the standalone model, where the simulation only has to produce a transcript as a whole, in the UC framework the simulator interacts continuously with an *environment*—typically denoted by $\mathcal{Z}$. This means that the simulator can, in particular, not rewind the environment. As such, the UC framework provides much stronger security guarantees than the standalone model, but comes with some restrictions. Without a trusted setup no protocol can securely UC-realize functionalities such as commitments [12], while computational constructions in the standalone model exist. Constructions in the UC framework also hold in the standalone model and, conversely, impossibilities in the standalone model extend to the UC framework.

The synchronous model. We assume a *synchronous* communication network. In particular, we require the following properties of the synchronous model, where we use the terms defined by Canetti [10, 2020 version, Section 7.3.3]:

Round Awareness. Each protocol description contains a maximum number of rounds. The round is monotonically increasing throughout the execution and only increases if all honest parties have been activated in this round. At each activation, honest parties are aware of the current round.

Guaranteed and Authentic Message Delivery. As mentioned before, we do not assume direct channels between parties and instead model communication between parties using setup functionalities. We require these setups to output their messages to the honest parties in a *guaranteed* and *authentic* way.

Note that in a synchronous model the environment still manages the order in which honest parties are activated.

Remark 2. The second requirement, Guaranteed and Authentic Message Delivery, does not imply *Guaranteed Output Delivery*: Each setup has its own special input and special output tape for interacting with honest parties. We follow the general convention and model these as inaccessible for the adversary; neither is it possible for the adversary to *suppress* the output, nor is it possible to *change* it in any way.

The only way an adversary can interact in this scenario is by *aborting* the setup, which requires a non-empty set of corrupted parties C and changes the originally intended output to $(\texttt{abort}, C)$; but this output cannot be suppressed or changed by the adversary.

This way of changing the output to $(\texttt{abort}, C)$, such that honest parties do not receive the original output of the setup, is why our requirements to not imply Guaranteed Output Delivery and thus evade the impossibility of fairness.

Remark 3. Assuming a synchronous model is necessary to prevent Denial-of-Service attacks because in an asynchronous model, the adversary can drop all messages [15, 6], resulting in a situation similar to (non-identifiable) abort. This renders Identifiable Abort essentially useless. The synchronous model, however, provides *guaranteed termination* [33]. When additionally assuming Identifiable Abort this means that the adversary can only either let the functionality terminate, or abort at the cost of revealing the identity of at least one malicious party.

Identifiable Abort. Unfortunately, fairness and thus guaranteed output is impossible against a dishonest majority [16]. On the other hand, the weaker notion of security with (non-identifiable) abort [34, 31], where the adversary can abort the protocol at any time without repercussions, leaves the protocol vulnerable to Denial-of-Service attacks.

To sidestep this issue we consider security with Identifiable Abort (IA) as formalized by Ishai, Ostrovsky, and Zikas [30] (see also [2, 29]). Here, abort is possible, but only by revealing the *same* identity of (at least) one malicious party to all participants. This property disincentivizes adversaries to cheat, especially if coupled with some form of penalty mechanism.

Definition 6 (Identifiable Abort [30]). *Let $n = n(\lambda)$ and let $\mathcal{F}^n$ be an ideal n-party functionality with n parties P and malicious parties $C \subsetneq P$. $\mathcal{F}^n$ has* ***(Multi-)Identifiable Abort****, iff on input* $(\texttt{abort}, C')$ *s.t.* $\emptyset \neq C' \subseteq C$ *from the adversary $\mathcal{F}^n$ sends* $(\texttt{abort}, C')$ *to all parties and terminates. $\mathcal{F}^n$ has Uni-Identifiable Abort, iff $\mathcal{F}^n$ has* Multi-IA *and* $|C'| = 1$.

Notation 9 (Functionalities with Identifiable Abort). *Let $n = n(\lambda)$. We denote by $\mathcal{F}^n$ an n-party functionality with Identifiable Abort (IA). In contrast, the original work [30] uses the notation $\mathcal{F}_{\perp}^{\mathsf{ID}}$. To prevent an overloaded notation we stick to $\mathcal{F}^n$ and always assume that all functionality enjoy security with IA.*

Note that functionalities with Identifiable Abort are not well-formed, meaning that they know which of the parties are corrupted and which are honest. This is inherently necessary to check whether $C' \subseteq C$.

Additional care has to be taken into the protocol design. We generally assume that the protocols and functionalities are *not* fair. This means, that the adversary can learn sensitive information in each protocol run, which it can leverage during the next execution. In our protocols, we mitigate this problem through the use of appropriate secret-sharings.

B.3 Functionalities / Setups

In this section, we introduce the ideal functionalities we use. We note that all following functionalities are defined with Identifiable Abort and are inherently *unfair*. It might be interesting to also consider fair variants to investigate the connection between fairness and Identifiable Abort for certain functionalities (for which fairness is not ruled out) in the vein of Cohen and Lindell [17].

Broadcast. First, we define a broadcast which is essentially the one from [14], though we only let parties broadcast messages to all parties not any subset of parties.

Functionality $\mathcal{F}_{\mathsf{BC}}^n$

$\mathcal{F}_{\mathsf{BC}}^n$ proceeds as follows, running with security parameter λ, $n = n(\lambda)$ parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\texttt{input}, m \in \{0,1\}^{\lambda})$ from party P_i, send $(\texttt{output}, \mathsf{P}_i, m)$ to $\mathcal{S}$. Upon the next activation, send $(\texttt{output}, \mathsf{P}_i, m)$ to all parties.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.

Commitments. Next, we define a (one-to-many) bit commitment, in the spirit of [14], adapted to the IA setting. We call the one-to-all commitment a global commitment.

Functionality $\mathcal{F}^n_{\mathsf{COM}}$

$\mathcal{F}^n_{\mathsf{COM}}$ proceeds as follows, running with security parameter λ, $n = n(\lambda)$ parties $P = \{\mathsf{S}, \mathsf{R}_1, ..., \mathsf{R}_{n-1}\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\mathtt{commit}, m \in \{0,1\}^\lambda)$ from party S, send $(\mathtt{receipt\ commit})$ to all parties. Ignore further messages $(\mathtt{commit}, \cdot)$ from S.
- When receiving $(\mathtt{open})$ from party S, send $(\mathtt{open}, m)$ to $\mathcal{S}$. Upon the next activation, send $(\mathtt{output}, m)$ to all parties and terminate.
- When receiving $(\mathtt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\mathtt{abort}, C')$ to all parties and terminate.

In the proof of Theorem 2 we use a one-to-one variant, which is essentially the same, except that only one fixed receiver obtains the opened value. We include it for completeness.

Functionality $\mathcal{F}^n_{\mathsf{COM},1:1}$

$\mathcal{F}^n_{\mathsf{COM},1:1}$ proceeds as follows, running with security parameter λ, $n = n(\lambda)$parties $P = \{\mathsf{S}, \mathsf{R}, ...\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\mathtt{commit}, m \in \{0,1\}^\lambda)$ from party S, send $(\mathtt{receipt\ commit})$ to all parties. Ignore further messages of the type $(\mathtt{commit}, \cdot)$ from S.
- When receiving $(\mathtt{open})$ from party S, send $(\mathtt{open}, \perp)$ to $\mathcal{S}$ (or $(\mathtt{open}, m)$ if R is corrupted). Upon the next activation, send $(\mathtt{open}, m)$ to R and send $(\mathtt{open}, \perp)$ to all parties, then terminate.
- When receiving $(\mathtt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\mathtt{abort}, C')$ to all parties and terminate.

SFE-complete functionalities. We start by providing a formal description of the functionality for Secure Function Evaluation.

Functionality $\mathcal{F}^n_{\mathsf{SFE},f}$

$\mathcal{F}^n_{\mathsf{SFE},f}$ proceeds as follows, running with security parameter λ, $n = n(\lambda)$ parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$, malicious parties $C \subseteq P$, adversary $\mathcal{S}$ and (possibly randomized) function $f\colon (x_1, ..., x_n) \mapsto (y_1, ..., y_n)$ with private input x_i and output y_i for P_i. Messages not covered here are ignored.

- When receiving $(\mathtt{input}, x_i)$ from P_i with $x_i \in \{0,1\}^\lambda$, store (i, x_i) and send $(\mathtt{receipt}, \mathsf{P}_i)$ to each corrupted party P_j and $\mathtt{receipt}$ to $\mathcal{S}$. Upon the next activation, send $(\mathtt{receipt}, \mathsf{P}_i)$ to each honest party P_j.
- When there are (i, x_i) stored for all $i \in [n]$, then send $(\mathtt{output}, y_j)$ to each corrupted party P_j and $(\mathtt{output})$ to $\mathcal{S}$. Upon the next activation, send $(\mathtt{output}, y_j)$ to each honest party P_j, then terminate.
- When receiving $(\mathtt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\mathtt{abort}, C')$ to all parties, and then terminate.

In particular, this functionality allows to instantiate the following functionality $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ where $f_\mathcal{D}$ ignores the inputs and samples from the distribution $\mathcal{D}$.

Functionality $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ adapted from [30]

$\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ proceeds as follows, running with security parameter λ, $n = n(\lambda)$ parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$, malicious parties $C \subseteq P$, adversary $\mathcal{A}$ and efficiently samplable distribution $\mathcal{D}$. Messages not covered here are ignored.

- When receiving `start` from P or $\mathcal{S}$, sample $(r_1, ..., r_n) \leftarrow \mathcal{D}$, output r_i to each corrupted party P_i. Upon the next activation, send r_i to each honest party P_j, then terminate.
- When receiving (`abort`, C') from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output (`abort`, C') to all parties, and then terminate.

Next, we formulate a variant of OT originally introduced as Verifiable OT [20], which was later described as Committed Oblivious Transfer [22]. We call our formalization as an ideal functionality Fully Committed Oblivious Transfer (FCOT), which extends classical OTs in three ways: 1. it includes $n-2$ witnesses, which obtain a *receipt* if the message has been transferred successfully, 2. the sender S is committed to both m_0 and m_1, and 3. the receiver R is committed to c.

Functionality $\mathcal{F}^n_{\mathsf{FCOT}}$

$\mathcal{F}^n_{\mathsf{FCOT}}$ proceeds as follows, running with security parameter λ, $n = n(\lambda)$ parties $P = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, ..., \mathsf{W}_{n-2}\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving (`messages`, $m^0, m^1 \in \{0,1\}^\lambda$) from S, store m^0, m^1 and send (`receipt messages`) to all parties and $\mathcal{S}$ if (`receipt choice`) has not been sent. Ignore further messages of the type (`messages`, $\cdot, \cdot$) from S.
- When receiving (`choice`, $c \in \{0,1\}$) from R, store c and send (`receipt choice`) to all parties and $\mathcal{S}$ if (`receipt messages`) has not been sent. Ignore further messages of the type (`choice`, $\cdot$) from R.
- When both m^0, m^1 and c are stored, send (`receipt transfer`) to $\mathcal{S}$ (or (`output`, m^c) if R is corrupted). Upon the next activation, send (`output`, m^c) to R, and (`receipt transfer`) to each party.
- When receiving (`open message`, $b \in \{0,1\}$) from S and m^0, m^1 are stored, send (`open message`, b, m_b) to $\mathcal{S}$. Upon the next activation, send (`open message`, b, m_b) and to each party. Ignore further messages (`open message`, b) from S.
- When receiving (`open choice`) from R and c is stored, send (`open choice`, c) to $\mathcal{S}$. Upon the next activation, send (`open choice`, c) and to each party. Ignore further messages from R.
- When receiving (`abort`, C') from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output (`abort`, C') to all parties and terminate.

Note that this functionality assigns dedicated roles to the participating parties. Because our results are modeled in the UC framework, multiple instances can be arbitrarily composed to allow OTs between any two parties.

Helper functionalities. Next, we define two helper functionalities whose purpose is to simplify our protocol descriptions. These variants of the $\mathcal{F}^n_{\mathsf{COM}}$ and

$\mathcal{F}^n_{\mathsf{FCOT}}$ functionalities take as inputs sharings of the message instead of the message resp. choice bit themselves. Since we study the question of the minimal complete cardinality (MCC), we assume setups of some size (here β) are given. Recall that for UC commitments setups are necessary. To determine the MCC, we do not care about how these setups are realized.

For notational ease, our functionalities allow each party to perform an OT with each other party, resp. each party can commit to all other parties. First, we define a Shared Oblivious Transfer (SOT) functionality which enables all participating parties to verify that the inputs encoded in the sharings are consistent with the inputs of some other (global) setup functionalities, in particular a global commitment of the shares. The notation $\mathsf{P}_i \to \mathsf{P}_j$ signifies an OT with sender P_i and receiver P_j.

Functionality $\mathcal{F}^\beta_{\mathsf{SOT}}$

$\mathcal{F}^\beta_{\mathsf{SOT}}$ proceeds as follows, running with security parameter λ, sharing parameter $\ell = \ell(\lambda)$, number of probing shares $\rho = \rho(\lambda)$, $\beta = \beta(\lambda)$ parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_\beta\}$, malicious parties $C \subseteq P$, and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\mathtt{messages}, \mathsf{P}_i \to \mathsf{P}_j, \mu^0, \mu^1 \in (\{0,1\}^\lambda)^{2\ell})$ s.t. $\forall b \in \{0,1\} : \text{Recover}_{\ell,2\ell}(\mu^b) \neq \perp$ from P_i, store $(\mathsf{P}_i \to \mathsf{P}_j, \mu^0, \mu^1)$ and send $(\mathtt{receipt\ messages}, \mathsf{P}_i \to \mathsf{P}_j)$ to $\mathcal{S}$. Upon the next activation, send $(\mathtt{receipt\ messages}, \mathsf{P}_i \to \mathsf{P}_j)$ to each party. Ignore further messages of this type from P_i to P_j.
- When receiving $(\mathtt{choice}, \mathsf{P}_i \to \mathsf{P}_j, \gamma \in (\{0,1\}^\lambda)^{2\ell})$ s.t. $\text{Recover}_{\ell,2\ell}(\gamma) \in \{0,1\}$ from P_j, store $(\mathsf{P}_i \to \mathsf{P}_j, \gamma)$ and send $(\mathtt{receipt\ choice}, \mathsf{P}_i \to \mathsf{P}_j)$ to $\mathcal{S}$. Upon the next activation, send $(\mathtt{receipt\ choice}, \mathsf{P}_i \to \mathsf{P}_j)$ to each party. Ignore further messages of this type from P_j to P_i.
- When both $(\mathsf{P}_i \to \mathsf{P}_j, \mu^0, \mu^1)$ and $(\mathsf{P}_i \to \mathsf{P}_j, \gamma)$ are stored, draw some "probing indices" $\nu \leftarrow \binom{[2\ell]}{\rho}$ and decode $c \leftarrow \text{Recover}_{\ell,2\ell}(\gamma)$.
  Send $(\mathtt{receipt\ transfer}, \mathsf{P}_i \to \mathsf{P}_j, \nu, \mu^0_\nu, \mu^1_\nu, \gamma_\nu)$ to $\mathcal{S}$ (additionally $(\mathtt{output}, \mathsf{P}_i \to \mathsf{P}_j, \mu^c)$ if P_j is corrupted). Upon the next activation, send $(\mathtt{receipt\ messages}, \mathsf{P}_i \to \mathsf{P}_j)$ to each party and $(\mathtt{output}, \mathsf{P}_i \to \mathsf{P}_j, \mu^c)$ to P_j.
- When receiving $(\mathtt{open\ message}, \mathsf{P}_i \to \mathsf{P}_j, b \in \{0,1\})$ from P_i and μ^0, μ^1 are stored, send $(\mathtt{open\ message}, \mathsf{P}_i \to \mathsf{P}_j, b, m^b)$ to $\mathcal{S}$ where $m^b \leftarrow \text{Recover}_{\ell,2\ell}(\mu^b)$. Upon the next activation, send $(\mathtt{open\ message}, \mathsf{P}_i \to \mathsf{P}_j, b, m^b)$ to all parties.
- When receiving $(\mathtt{open\ choice}, \mathsf{P}_i \to \mathsf{P}_j)$ from P_j and γ is stored, send $(\mathtt{open\ choice}, \mathsf{P}_i \to \mathsf{P}_j, c)$ to $\mathcal{S}$ where $c \leftarrow \text{Recover}_{\ell,2\ell}(\gamma)$. Upon the next activation, send $(\mathtt{open\ choice}, \mathsf{P}_i \to \mathsf{P}_j, c)$ to all parties.
- When receiving $(\mathtt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\mathtt{abort}, C')$ to all parties and terminate.

Next, we define a Shared Commitment (SCOM) functionality that enables the verification of consistency of inputs across setups. The motivation behind this functionality is that it admits a standard commit-then-prove technique in our construction. The idea is to *commit* the same sharing into many (different) setups and *prove* the equality (of the encoded message) by opening (sufficiently many) random shares. In this way a commitment to few parties can be extended to many parties.

Functionality $\mathcal{F}^{\beta}_{\mathsf{SCOM}}$

$\mathcal{F}^{\beta}_{\mathsf{SCOM}}$ proceeds as follows, running with security parameter λ, sharing parameter $\ell = \ell(\lambda)$, $\beta = \beta(\lambda)$ parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_\beta\}$, malicious parties $C \subseteq P$, and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\texttt{commit}, \mu^i \in (\{0,1\}^\lambda)^{2\ell})$ from P_i s.t. $\text{Recover}_{\ell,2\ell}(\mu^i) \neq \bot$, store μ^i and send $(\texttt{receipt commit}, \mathsf{P}_i)$ to $\mathcal{S}$. Upon the next activation, send $(\texttt{receipt commit}, \mathsf{P}_i)$ to each party. Ignore further messages $(\texttt{input}, \cdot)$ from P_i.
- When receiving $(\texttt{open}, U \subseteq [2\ell])$ from P_i, if the sharing μ^i is stored, send $(\texttt{output}, \mathsf{P}_i, U, \mu^i_U)$ to $\mathcal{S}$. Upon the next activation, send $(\texttt{output}, \mathsf{P}_i, U, \mu^i_U)$ to each party.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.

For our proof we define the two-party functionality Secure Message Transfer (SMT) which usually describes the underlying communication model. Our construction do not explicitly require pairwise channels between parties, yet we still use the $\mathcal{F}^2_{\mathsf{SMT}}$ to simplify our protocols. This does not change the statements w.r.t. the minimal complete cardinality.

Functionality $\mathcal{F}^2_{\mathsf{SMT}}$

$\mathcal{F}^n_{\mathsf{SMT}}$ proceeds as follows, running with security parameter λ, parties $P = \{\mathsf{P}_1, \mathsf{P}_2\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\texttt{input}, m \in \{0,1\}^\lambda)$ from party P_1, send m to P_2.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.

C Trust Graph: Identification via Conflicts

Let us start with the formal definition of the TG.

Definition 7 (Trust Graph). *The* Trust Graph *(TG) is defined as the output of the following functionality.*

Functionality $\mathcal{F}^n_{\mathsf{TG}}$

$\mathcal{F}^n_{\mathsf{TG}}$ proceeds as follows, running with parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- *Upon first activation, initiate the set of conflict edges $E := \binom{P}{\leq 2}$.*
- *When receiving a message $(\texttt{conflict}, \mathsf{P}_i \in P)$ from P_j, for $i \neq j$ remove the edge $\{\mathsf{P}_i, \mathsf{P}_j\}$ from the set of edges E and send $(\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i)$ to the adversary.*
- *When receiving a message $(\texttt{conflict}, P' \subseteq P)$ from P_j, for each $\mathsf{P}_i \in P'$ s.t. $i \neq j$ remove the edge $\{\mathsf{P}_i, \mathsf{P}_j\}$ from E and send $(\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i)$ to the adversary.*[a]
- *When receiving a message $(\texttt{query})$ from P_j, output the* Trust Graph *$G := (P, E)$ to P_j.*
- *When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.*

[a] This instruction is stated merely for notational ease of conflicts with multiple parties.

The object of the Trust Graph $G := (P, E)$ itself is an undirected reflexive graph on all parties returned by the $\mathcal{F}^n_{\mathsf{TG}}$ functionality.

This functionality internally maintains the graph. Each time a party declares a conflict with another party, the corresponding edge is removed from the graph. In a given round each party—upon query—the $\mathcal{F}^n_{\mathsf{TG}}$ returns the *same* graph. This implies a *consistent* view to all parties for any protocol π_{TG} that realizes $\mathcal{F}^n_{\mathsf{TG}}$. In particular, the $\mathcal{F}^n_{\mathsf{TG}}$ can be realized using only broadcast.

Lemma 6. *Let $n = n(\lambda)$. There is an efficient protocol π_{TG} that perfectly securely UC-realizes $\mathcal{F}^n_{\mathsf{TG}}$ in the $\{\mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against environments that (maliciously) corrupt up to n parties. Formally,*

$$\mathcal{F}^n_{\mathsf{BC}} \overset{\text{perf}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{TG}} \ . \tag{16}$$

We include the proof in Appendix E. To link the abort condition of a protocol to the disconnectivity of the TG we want the protocol to ensure that honest parties are never in conflict. This is in particular fulfilled by the following definition. Put simply, the following rules say that the parties should 1) query the TG from $\mathcal{F}^n_{\mathsf{TG}}$, 2) modify the TG according to aborted setups or an abort of the protocol itself, and 3) report all new conflicts back to $\mathcal{F}^n_{\mathsf{TG}}$.

Definition 8 (Abort-respecting protocols). *A protocol with parties P and corrupted parties $C \subseteq P$ in some F-hybrid model s.t. $\mathcal{F}^n_{\mathsf{TG}} \in F$ is* abort-respecting,[15] *iff each honest party $\mathsf{P} \in H$ obeys the following ruleset. P maintains three sets of parties: $\widehat{P} := \emptyset$ is the set of currently "feuding"[16] parties, $\widehat{C} := \emptyset$ is*

[15] [30] also use the term *abort-respecting*, we adapt it to the context of Trust Graphs.

[16] The parties just that witnessed the abort of a setup are "feuding" because they declare conflicts to eventually disconnect the corresponding subgraph.

the set of initially accused (identified) parties after a setup-abort, and $\widehat{D} \coloneqq \emptyset$ as the set of parties that P *has currently identified as malicious.*

Rule 1 *At the onset of each round* P *sends* $(\mathtt{query})$ *to* $\mathcal{F}^n_{\mathsf{TG}}$ *to obtain* $G \coloneqq (P, E)$ *in the next round.*

Rule 2 *When* P *reads* $(\mathtt{abort}, C')$ *where* $\emptyset \neq C' \subseteq C \cap P'$ *as output from some setup on parties* $P' \ni \mathsf{P}$*, then* P *inputs* $(\mathtt{conflict}, C')$ *into* $\mathcal{F}^n_{\mathsf{TG}}$ *and defers processing all other inputs until the subgraph on* P' *is disconnected. Then* P *sets* $\widehat{P} = P'$ *and* $\widehat{C} = C'$*, and continues to Rule* 3*.*

Rule 3 *If* $\widehat{C} \neq \emptyset$*,* P *sets* $\widehat{D} \coloneqq \widehat{D} \cup \widehat{C}$*, and otherwise skips this rule. The party* P *extracts the set* L *of "loyal" parties in* $\widehat{P}$ *that are not in conflict with all parties in* $\widehat{D}$ *and send* $(\mathtt{conflict}, L)$ *to* $\mathcal{F}^n_{\mathsf{TG}}$*, then* $\hat{D} \coloneqq \hat{D} \cup L$*. This process is repeated in each following round (only applying Rule* 3 *and deferring all other inputs) until the subgraph on* $\widehat{P}$ *is disconnected. Once the subgraph on* $\widehat{P}$ *is disconnected* P *resets* $\widehat{C} = \widehat{P} = \emptyset$ *and continues according to the protocol.*

Rule 4 *After obtaining* $G \coloneqq (P, E)$ *from* $\mathcal{F}^n_{\mathsf{TG}}$*, the party* P *checks if* G *is disconnected. If so,* P *determines its connected component* $P' \ni \mathsf{P}$ *(e.g. via breadth-first-search) and aborts the computation with* $(\mathtt{abort}, P \setminus P')$*.*
Otherwise, P *checks if there are parties with more than* t *conflicts. If so,* P *sets* $\widehat{C}$ *as the set of parties with more than* t *conflicts, and* $\widehat{P} = P$*, and continues according to Rule* 3*.*

Rule 5 *Whenever* P *is about to abort the computation with output* $(\mathtt{abort}, C')$ *s.t.* $\emptyset \neq C' \subseteq C$ *and the overall TG is (still) connected,* P *sets* $\widehat{C} = C'$ *and* $\widehat{P} = P$*, and continues according to Rule* 3*.*[17]

Rule 6 *The party* P *only declares conflicts with malicious parties. Conflicts declared according to the rules above are called* generic*, any other conflicts are called* specific*.*

First, we want to discuss Rule 6, i.e., there are no honest-honest conflicts. In particular, generic conflicts fulfill this requirement honest parties only declare a conflicts

1. with parties identified in the abort of a setup or the protocol itself, or
2. with parties who stay loyal to such identified parties, or
3. with parties that have more conflicts than are allowed.

First, because the setups and the protocol are secure with Identifiable Abort by assumption, all identified parties upon abort (of a setup) must be malicious with overwhelming probability. Second, by Rule 2 honest parties do not stay loyal to identified parties. Third, if a party has strictly more than t conflicts it must be malicious. For contradiction, suppose the following: In a protocol any honest party P gets more than t conflicts for the first time in some fixed round ρ. Consequently, all (strictly more than t) parties in conflict with P must be malicious because before round ρ conflicts were only issued by P due to Rule

[17] If the underlying protocol is secure with Identifiable Abort, then all honest parties will abort with the same identified parties in the same round, hence $\widehat{C}$ and $\widehat{P}$ are consistent among all honest parties.

2 and Rule 5. This, however, contradicts the assumption that at most t parties are malicious, hence P must be malicious. In summary, any conflict contains at least one malicious party.
Moreover, the process of gradual conflict declarations terminates within at most n rounds because in each non-final round at least one loyalist is moved into the set of malicious parties $\widehat{D}$.

We want to elaborate on the notion of *specific* conflicts. A specific conflict must—by design of the protocol—ensure that it contains at least one malicious party. That is, whenever presenting an abort-respecting protocol, one must prove its specific conflict rules indeed cause no honest-honest conflicts. We do this for our construction in Appendix D.2.
We emphasize that Definition 8 does not dictate which specific conflicts *are* declared in a protocol but it dictates which ones *may not* be declared, namely any conflict that could result in an honest-honest conflict. For example, consider a protocol where each party broadcasts `start` in the first round. If any party P instead broadcasts $\perp$ or nothing, then P must be malicious and it is safe to declare a (specific) conflict with P because P violated the protocol specification in an obvious manner.[18] Looking ahead to the next lemma, we note that a protocol can canonically be made abort-respecting by amending the protocol with instructions (Rule 1, Rule 2, Rule 5 and Rule 4) corresponding to generic conflicts but none for specific conflicts.

Now that we have a good intuition of the presented graph properties, we can formally show that the disconnectivity of the Trust Graph corresponds to the honest parties' ability for Identifiable Abort.

Corollary 5 (Identifiable Abort $\iff$ disconnected Trust Graph). *Let $n = n(\lambda)$ and let F' be a set of setups for at most n parties. Let π^F be an* abort-respecting *protocol that securely UC-realizes a functionality $\mathcal{F}^n$ with Identifiable Abort in the F-hybrid model where $F := F' \cup \{\mathcal{F}^n_{\mathsf{TG}}\}$.*

1. *If the TG of π^F becomes disconnected, the (honest) parties abort by identifying malicious parties.*
2. *Upon abort, the TG G of π^F must be disconnected.*

Proof. The first statement follows directly from Rule 4 in Definition 8. Let P be the set of parties. Note that all honest parties are in the same connected component $\emptyset \neq P' \subset P$ of the TG. Hence, all parties in $P \setminus P'$ must be malicious. As such, an honest party in P' aborts correctly with $P \setminus P'$. The second statement follows from Rule 5 in Definition 8. $\square$

Lemma 7 (TG augmentation). *Let n be the number of parties P of which at most $0 \leq t \leq n$ are malicious. Let π^F be a protocol that securely UC-realizes a functionality $\mathcal{F}^n$ in some model $F \not\ni \mathcal{F}^n_{\mathsf{TG}}$. Denote by $\widetilde{\pi}^{F,\mathcal{F}^n_{\mathsf{TG}}}$ the $F \cup \{\mathcal{F}^n_{\mathsf{TG}}\}$-hybrid protocol that is identical to π^F with the addition that it uses $\mathcal{F}^n_{\mathsf{TG}}$ in an* abort-respecting *way according to Definition 8. Then $\widetilde{\pi}^{F,\mathcal{F}^n_{\mathsf{TG}}}$ also securely UC-realizes $\mathcal{F}^n$, i.e., $\widetilde{\pi}^{F,\mathcal{F}^n_{\mathsf{TG}}} \geq \mathcal{F}^n$.*

[18] This is similar to the usage of the Trust Graph in [42].

Proof. Here, the intuition is that the environment can infer the TG from its own behavior without querying $\mathcal{F}^n_{\mathsf{TG}}$. Furthermore, the additional behavior described in Definition 8 is deterministic and can be inferred from a transcript of the base protocol π^F by adding the appropriate calls to $\mathcal{F}^n_{\mathsf{TG}}$ and potentially aborting, i.e., ending the protocol with abort-messages, if Rule 4 in Definition 8 occurs.

Now, we formally prove the statement. First, note that $\exists \mathcal{S}\, \forall \mathcal{Z} : \mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}(\mathcal{Z}) \approx \mathsf{IDEAL}_{\mathcal{F}^n, \mathcal{S}}(\mathcal{Z})$ because π^F securely UC-realizes $\mathcal{F}^n$.[19] That is, $\mathcal{S}$ is the simulator for protocol π^F.
Let $\widetilde{\mathcal{Z}}$ denote an environment for protocol $\widetilde{\pi}^{F, \mathcal{F}^n_{\mathsf{TG}}}$. Let $[\widetilde{\mathcal{Z}}]$ be the environment for protocol π^F that internally emulates $\widetilde{\mathcal{Z}}$. As the inner environment $\widetilde{\mathcal{Z}}$ expects an interface to $\mathcal{F}^n_{\mathsf{TG}}$ the outer environment $[\widetilde{\mathcal{Z}}]$ simulates a $\mathcal{F}^n_{\mathsf{TG}}$ setup for it. $[\widetilde{\mathcal{Z}}]$ outputs whatever $\widetilde{\mathcal{Z}}$ outputs with the following modification: $[\widetilde{\mathcal{Z}}]$ also simulates the additional behavior of honest parties as described in Definition 8 using the internally simulated $\mathcal{F}^n_{\mathsf{TG}}$. As such, the output transcript of $[\widetilde{\mathcal{Z}}]$ when running with π^F is identical to the output of $\widetilde{\mathcal{Z}}$ when playing with $\widetilde{\pi}^{F, \mathcal{F}^n_{\mathsf{TG}}}$, i.e.,

$$\mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}\left(\left[\widetilde{\mathcal{Z}}\right]\right) \equiv \mathsf{REAL}_{\widetilde{\pi}^{F, \mathcal{F}^n_{\mathsf{TG}}}, \mathcal{A}_\mathcal{D}}\left(\widetilde{\mathcal{Z}}\right) . \tag{17}$$

By the security of π^F there exists some simulator $\widetilde{\mathcal{S}}$ s.t.

$$\mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}\left(\left[\widetilde{\mathcal{Z}}\right]\right) \approx \mathsf{IDEAL}_{\mathcal{F}^n, \widetilde{\mathcal{S}}}\left(\left[\widetilde{\mathcal{Z}}\right]\right) . \tag{18}$$

Furthermore, let $[\widetilde{\mathcal{S}}]$ be the simulator that internally emulates $\widetilde{\mathcal{S}}$ and also simulates an $\mathcal{F}^n_{\mathsf{TG}}$ setup. Analogously to $[\widetilde{\mathcal{Z}}]$, the outer simulator $[\widetilde{\mathcal{S}}]$ outputs whatever the inner simulator $\widetilde{\mathcal{S}}$ outputs with the modification that $[\widetilde{\mathcal{S}}]$ also simulates the additional behavior of honest parties as described in Definition 8 using the internally simulated $\mathcal{F}^n_{\mathsf{TG}}$, hence

$$\mathsf{IDEAL}_{\mathcal{F}^n, \widetilde{\mathcal{S}}}\left(\left[\widetilde{\mathcal{Z}}\right]\right) \equiv \mathsf{IDEAL}_{\mathcal{F}^n, \left[\widetilde{\mathcal{S}}\right]}\left(\widetilde{\mathcal{Z}}\right) . \tag{19}$$

We can conclude that for any $\widetilde{\mathcal{Z}}$

$$\begin{aligned} \mathsf{REAL}_{\widetilde{\pi}^{F, \mathcal{F}^n_{\mathsf{TG}}}, \mathcal{A}_\mathcal{D}}\left(\widetilde{\mathcal{Z}}\right) &\equiv \mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}\left(\left[\widetilde{\mathcal{Z}}\right]\right) \\ \approx \mathsf{IDEAL}_{\mathcal{F}^n, \widetilde{\mathcal{S}}}\left(\left[\widetilde{\mathcal{Z}}\right]\right) &\equiv \mathsf{IDEAL}_{\mathcal{F}^n, \left[\widetilde{\mathcal{S}}\right]}\left(\widetilde{\mathcal{Z}}\right) . \end{aligned} \tag{20}$$

□

D Full Constructions & Proofs

D.1 Impossibility / Lower Bound

In this section we formally prove our impossibility result as sketched in Section 1.3.

[19] As in Notation 2, $\mathcal{A}_\mathcal{D}$ is the canonical dummy adversary.

Lemma 2 (Connected graph $\implies$ many disconnected $\beta-1$-subgraphs). *Let $n, t \in \mathbb{N}$ s.t. $n/2 \leq t \leq n-1$, and let $\beta := \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$. Furthermore, let V be a set of n vertices and let $v, v' \in V : v \neq v'$ be two different vertices. There exist some edges $E \subseteq \binom{V}{\leq 2}$ s.t.*

1. $G := (V, E)$ is an undirected, reflexive and connected *graph,*
2. $\forall \{u, u'\} \in E : |\,\mathrm{N}_G(u) \cap \mathrm{N}_G(u')| \geq n - t$,
3. for each $V' \in M$ the subgraph $G' := (V', E \cap \binom{V'}{\leq 2})$ is disconnected
where

$$N_u := \begin{cases} \{u\} & \text{, if } t = n-1 \\ \mathrm{N}_G(u) & \text{, else} \end{cases}$$

is the set of "effective" neighbors of any vertex u, and

$$M := \{V' \subseteq V \mid \quad V' \cap N_v \neq \emptyset \quad \wedge \quad V' \cap N_{v'} \neq \emptyset \quad \wedge \quad |V'| < \beta\}$$

is the set of relevant subsets of vertices that contain both an effective neighbor of v and an effective neighbor of v'.

Proof. The proof[20] itself is—by its graph-theoretical nature—quite technical, so we want to give a brief intuition first. To start, we give a constructive description of the graph G. This might be best visualized with Fig. 1; it depicts examples of the graph G for some parameters. Its structure looks somewhat like a caterpillar path but with triangles. After presenting the description of the graph, we prove all required properties. The key point to showing that if all *relevant* subgraphs (on vertex subset in M) are disconnected, then all such subgraphs contain vertices near each end of the caterpillar. Because the subgraphs are sufficiently small, some part in the middle of the caterpillar must be missing—dissecting the caterpillar.

Let $h := n-t$, let $V := \{v_1, ..., v_n\}$ and w.l.o.g. $v := v_1$ and $v' := v_n$. We define S- and L-segments by $S_j := \{v_{jh+2}, ..., v_{(j+1)h}\}$ and $L_j := \{v_{jh+1}\}$. There are $\lceil n/h \rceil$ many L-segments, and $\lceil (n-1)/h \rceil$ many S-segments if $h \geq 2$. Otherwise there are no S-segments, i.e., in case $h = 1$ the S-segments S_j are empty. Based on these segments we define the graph $G := (V, E)$ by

$$E := \{\{v_j, v_{j+1}\} \mid j \in [n-1]\} \cup \{\{v_j, v_j\} = \{v_j\} \mid j \in [n]\} \tag{21}$$

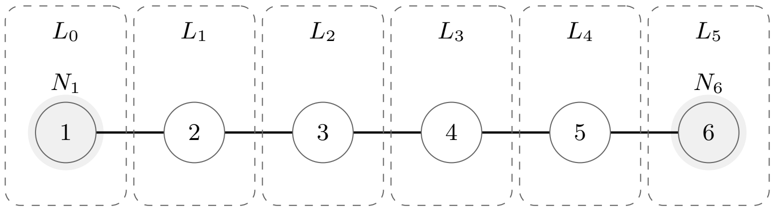
if $h = 1$ and

$$E := \left\{ \{u, u'\} \;\middle|\; \begin{array}{ll} \exists j \in [0, \lceil n/h \rceil - 1] : & u \in S_j \wedge u' \in S_j \cup L_j \cup L_{j+1} \\ \vee & u \in S_{\lfloor n/h \rfloor - 1} \wedge u' \in S_{\lfloor n/h \rfloor} \\ \vee & u = u' \end{array} \right\} \tag{22}$$

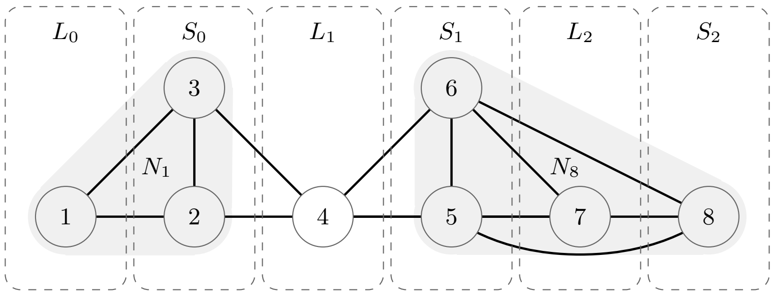
if $h \geq 2$. Obviously, G is reflexive. We show the following properties of G:
1. G is connected,
2. $\forall \{u, u'\} \in E : |\,\mathrm{N}_G(u) \cap \mathrm{N}_G(u')| \geq h$,
3. each subgraph G' on $V' \in M$ is disconnected.

To show Property 1 we notice that all vertices within a given segment are connected in G, i.e., $\binom{S_j}{\leq 2} \subseteq E$ and $\binom{L_j}{\leq 2} \subseteq E$. Furthermore, consecutive segments

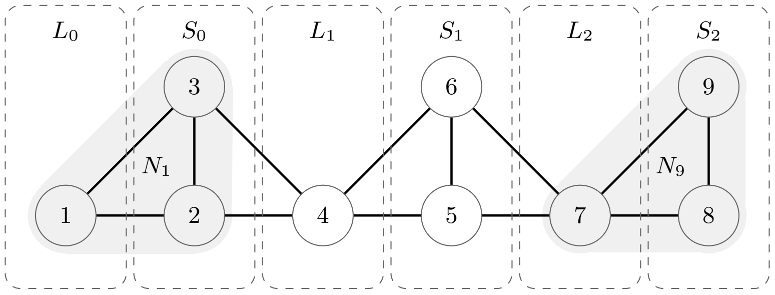
[20] This proof is very similar to the one of Claim 3.1 in [42].



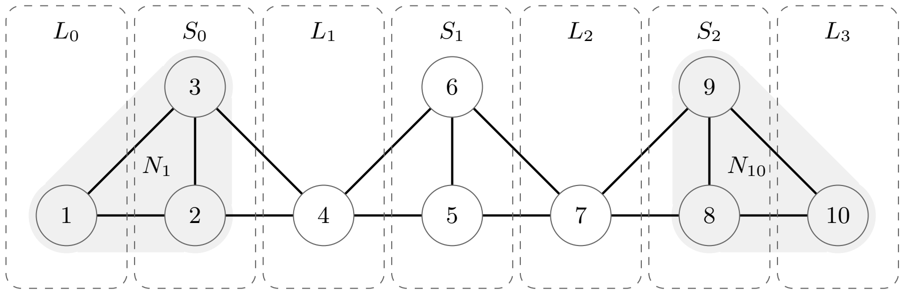
(a) Example of the graph G from Lemma 2 with $t = 5$, $n = 6$ and $\beta = 6$.



(b) Example of the graph G from Lemma 2 with $t = 5$, $n = 8$ and $\beta = 3$.



(c) Example of the graph G from Lemma 2 with $t = 6$, $n = 9$ and $\beta = 4$.



(d) Example of the graph G from Lemma 2 with $t = 7$, $n = 10$ and $\beta = 5$.

Fig. 1: Examples of the graphs from Lemma 2 with $V = [n]$, $v = 1$ and $v' = n$. Segments are represented as dashed boxes, and "effective neighbors" of v and v' in grey. Note that any shortest path from N_v to $N_{v'}$ (grey to grey) always has exactly β nodes. The graph structure is the same as the "multi-layer graph" in Figure 1 of [42].

are also connected in G. For $h = 1$ this is trivial, for $h \geq 2$ this is ensured by the condition $u \in S_j \wedge u' \in S_j \uplus L_j \uplus L_{j+1}$ in the definition of E.

Next, we show Property 3, i.e., for each $V' \in M$ the subgraph $(V', E \cap \binom{V'}{\leq 2})$ is disconnected. To this end, we prove that there exists a *splitting index* $j_{V'}$ which defines two partitions

$$X \coloneqq V' \cap \left\{v_1, ..., v_{j_{V'}h+1}\right\} \quad \text{and} \quad Y \coloneqq V' \cap \left\{v_{j_{V'}h+2}, ..., v_n\right\} \tag{23}$$

s.t. $L_{j_{V'}} \cap V' = \emptyset$ or $S_{j_{V'}} \cap V' = \emptyset$. In other words, V' is missing all vertices from some segment (in the middle of the caterpillar graph).
In the case $h = 1 \iff \beta = n$ we find for each $V' = V \setminus \{v_j\} = V \setminus L_j$ s.t. $1 < j < n$ that the splitting index is $j_{V'} = j - 1$. Because the overall graph G is a line graph, the strict subgraph on V' is disconnected.

Henceforth, we consider $h \geq 2 \iff \beta = \lceil n/h \rceil + \lfloor n/h \rfloor - 2$. Define for each $V' \in M$ the sets of possible splitting indices for the L- and S- segments respectively

$$J^{\mathrm{L}}_{V'} \coloneqq \{j_{\mathrm{L}} \in [1, \lceil n/h \rceil - 2] \mid L_{j_{\mathrm{L}}} \cap V' = \emptyset\} \tag{24}$$

and

$$J^{\mathrm{S}}_{V'} \coloneqq \{j_{\mathrm{S}} \in [1, \lfloor n/h \rfloor - 2] \mid S_{j_{\mathrm{S}}} \cap V' = \emptyset\} \ . \tag{25}$$

These sets describe which L-/S-segments are missing in V'. Note that $N_{v_1} = \{v_1, ..., v_h\}$ and $N_{v_n} \subseteq \{v_{(\lceil n/h \rceil - 2)h+2}, ..., v_n\}$ which implies

$$\forall j_{\mathrm{L}} \in [1, \lceil n/h \rceil - 2] : L_{j_{\mathrm{L}}} \cap (N_{v_1} \uplus N_{v_n}) = \emptyset \tag{26}$$

and

$$\forall j_{\mathrm{S}} \in [1, \lfloor n/h \rfloor - 2] : S_{j_{\mathrm{S}}} \cap (N_{v_1} \uplus N_{v_n}) = \emptyset \ . \tag{27}$$

For intuition, note that N_{v_1} and N_{v_n} are the "ends" of the caterpillar graph depicted in Fig. 1. The segments corresponding to $j_{\mathrm{L}} \in [1, \lceil n/h \rceil - 2]$ and $j_{\mathrm{S}} \in [1, \lfloor n/h \rfloor - 2]$ in the middle of the graph are disjoint from the ends.

Next, we show for each $V' \in M$ that $|J^{\mathrm{L}}_{V'} \cup J^{\mathrm{S}}_{V'}| > 0$, i.e., a splitting index exists. Suppose for contraction that no such index existed, i.e.,

$$\forall j_{\mathrm{L}} \in [1, \lceil n/h \rceil - 2] : L_{j_{\mathrm{L}}} \cap V' \neq \emptyset \iff |L_{j_{\mathrm{L}}} \cap V'| \geq 1 \tag{28}$$

and

$$\forall j_{\mathrm{S}} \in [1, \lfloor n/h \rfloor - 2] : S_{j_{\mathrm{S}}} \cap V' \neq \emptyset \iff |S_{j_{\mathrm{S}}} \cap V'| \geq 1 \ . \tag{29}$$

Moreover, we know that $N_{v_1} \cap V' \neq \emptyset$ and $N_{v_n} \cap V' \neq \emptyset$ by the definition of $M \ni V'$, hence $|(N_{v_1} \uplus N_{v_n}) \cap V'| \geq 2$. Then we can write V' as unions of disjoint subsets

$$V' \supseteq ((N_{v_1} \uplus N_{v_n}) \cap V') \uplus \left(\biguplus_{j_{\mathrm{L}}=1}^{\lceil n/h \rceil - 2} L_{j_{\mathrm{L}}} \cap V' \right) \uplus \left(\biguplus_{j_{\mathrm{S}}=1}^{\lfloor n/h \rfloor - 2} S_{j_{\mathrm{S}}} \cap V' \right) \tag{30}$$

which implies

$$|V'| \geq 2 + (\lceil n/h \rceil - 2) + (\lfloor n/h \rfloor - 2) = \lceil n/h \rceil + \lfloor n/h \rfloor - 2 = \beta \tag{31}$$

which contradicts the assumption $|V'| < \beta$. Now, we have shown that such a $j_{V'}$ exists. For the sake of simplicity we choose the minimal $j_{V'} \coloneqq \min(J^{\mathrm{L}}_{V'} \cup J^{\mathrm{S}}_{V'})$.

We go on to show that no vertex in X is connected to any vertex in Y in the graph G. Suppose for contradiction that $\{v_a, v_b\} \in E$ for some $v_a \in X \subset V'$ and $v_b \in Y \subset V'$, i.e., $a \leq j_{V'}h + 1$ and $b > j_{V'}h + 1$. Because v_a and v_b are connected in G, they must be in consecutive segments by the definition of E. Consequently, we find $v_a \in L_{j_{V'}}$ and $v_b \in S_{j_{V'}}$ because $a \leq j_{V'}h + 1$ and $b > j_{V'}h + 1$ and v_a and v_b are in consecutive segments. This directly contradicts the splitting property $L_{j_{V'}} \cap V' = \emptyset \vee S_{j_{V'}} \cap V' = \emptyset$.

Lastly, we show Property 2, i.e., that any pair of connected parties has at least h common neighbors. For $h = 1$ this is trivial. Henceforth consider $h \geq 2$. Note that for S-segments it holds that $j < \lfloor n/h \rfloor \implies |S_j| = h - 1$. For each $u, u' \in L_j \uplus S_j$ s.t. $j < \lfloor n/h \rfloor$ it holds that $\mathrm{N}_G(u), \mathrm{N}_G(u') \supseteq L_j \uplus S_j$, hence

$$\begin{aligned}&\mathrm{N}_G(u) \cap \mathrm{N}_G(u') \supseteq L_j \uplus S_j \implies \\ &\qquad |\mathrm{N}_G(u) \cap \mathrm{N}_G(u')| \geq |L_j \uplus S_j| = |S_j| + |L_j| = h - 1 + 1 = h\ . \qquad (32)\end{aligned}$$

For each $u, u' \in L_j \uplus S_j$ with $j = \lfloor n/h \rfloor$ it holds that $\mathrm{N}_G(u), \mathrm{N}_G(u') \supseteq S_{j-1} \uplus L_j$, hence

$$\begin{aligned}&\mathrm{N}_G(u) \cap \mathrm{N}_G(u') \supseteq S_{j-1} \uplus L_j \implies \\ &\qquad |\mathrm{N}_G(u) \cap \mathrm{N}_G(u')| \geq |S_{j-1} \uplus L_j| = |S_{j-1}| + |L_j| \geq h - 1 + 1 = h\ . \qquad (33)\end{aligned}$$

For each $u \in S_j, u' \in L_{j+1}$ with $j < \lceil n/h \rceil - 1$ it holds that $\mathrm{N}_G(u), \mathrm{N}_G(u') \supseteq S_j \uplus L_{j+1}$, hence

$$\begin{aligned}&\mathrm{N}_G(u) \cap \mathrm{N}_G(u') \supseteq S_j \uplus L_{j+1} \implies \\ &\qquad |\mathrm{N}_G(u) \cap \mathrm{N}_G(u')| \geq |S_j \uplus L_{j+1}| = |S_j| + |L_{j+1}| = h - 1 + 1 = h\ . \qquad (34)\end{aligned}$$

This concludes the proof. □

Theorem 2 (No transmitted commitment). *Let $n = n(\lambda), t = t(\lambda), \beta := \min(n, \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2)$ s.t. $n/2 \leq t < n$ and $\binom{n}{\beta} \in \mathrm{poly}(\lambda)$. No $\{\mathcal{F}^2, ..., \mathcal{F}^{\beta-1}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid protocol can securely UC-realize $\mathcal{F}^n_{\mathsf{COM},1:1}$ against environments that (maliciously) corrupt up to t parties. Formally, we get*

$$\left\{\mathcal{F}^2, ..., \mathcal{F}^{\beta-1}, \mathcal{F}^n_{\mathsf{BC}}\right\} \overset{\text{comp}}{\not\rightarrow}_t \mathcal{F}^n_{\mathsf{COM},1:1} \qquad (1)$$

where $\mathcal{F}^2, ..., \mathcal{F}^{\beta-1}$ stand for arbitrary functionalities of the respective cardinality, and $\mathcal{F}^n_{\mathsf{COM},1:1}$ is defined in Appendix B.3.
Consequently, the correlation complexity for UC-secure ID-MPC is at least β.

Proof. On a high level the proof proceeds similar to the the impossibility of UC commitments without setup in [12]. There an environment $\mathcal{Z}^0$ corrupts the sender but simulates the sender honestly, i.e., according to the protocol specification with input m. Because all parties act honestly, the receiver must eventually output the message m upon opening. However, in the ideal execution, this means that the simulator must extract the message m from the corrupted (but honestly acting) sender's message to the receiver *during* the commitment phase. Intuitively, this contradicts the hiding property the commitment. More formally, this simulator (or extractor) can be used by a second environment that corrupts the receiver, to extract the message m *before* its opening. This, however, is impossibile for a simulator to achieve because in the ideal execution the simulator has no information about the honest sender's message before obtaining it from the ideal commitment functionality upon opening.

In this proof we proceed similarly but we have to account for the fact that setups are aborted, hence not all parties can behave objectively honestly, i.e., according to the protocol. We remedy this by providing the corrupted (simulated) parties with alternative (faux) abort messages, that look legitimate to them. Thus, the simulated parties act honestly relative to their view of the Trust Graph.

Denote the set of parties by $P = \{\mathsf{P}_1 = \mathsf{S}, \mathsf{P}_2, ..., \mathsf{P}_n = \mathsf{R}\}$ where S is the sender and R is the receiver. Let $h := n - t$, and let $N_1 := \{\mathsf{P}_1, ..., \mathsf{P}_h\}$, $N_n := \{\mathsf{P}_{n-h+1}, ..., \mathsf{P}_n\}$ and $I := \{\mathsf{P}_{h+1}, ..., \mathsf{P}_{n-h}\}$ be three partitions of the parties. (Intuitively N_1 will be the neighbors of the sender and N_n will be the neighbors of the receiver.) Furthermore, denote by $\eta_{\texttt{commit}}$ the round number in which the last (honest) party outputs ($\texttt{receipt commit}$). Denote by $\text{ext}_{\mathsf{P}}^{\texttt{commit}} = (\eta, \iota, i, s_{\eta,\iota,i})_{\eta \leq \eta_{\texttt{commit}}, \iota, i}$ the list of all messages that party $\mathsf{P} \in P$ sends to any setup (excluding broadcast) during the commit phase. Here $s_{\eta,\iota,i}$ denotes the i-th message that P sends to the setup with session id ι in round η. For any set of parties $P' \subseteq P$ let $\text{ext}_{P'}^{\texttt{commit}} := (\text{ext}_{\mathsf{P}}^{\texttt{commit}})_{\mathsf{P} \in P'}$. Moreover, let $B^{\texttt{commit}} := (\eta, i, w_{\eta,\mathsf{P},i})_{\eta \leq \eta_{\texttt{commit}}, \mathsf{P}, i}$ denote all messages that were broadcasted during the commit phase.

We will prove the impossibility via a series of hybrid games. Suppose for contradiction[21] that $\{\mathcal{F}^{\beta-1}, \mathcal{F}_{\mathsf{BC}}^n\} \overset{\text{comp}}{\rightsquigarrow_t} \mathcal{F}_{\mathsf{COM},1:1}^n$, i.e., there exists a protocol $\pi^{\mathcal{F}^{\beta-1}, \mathcal{F}_{\mathsf{BC}}^n}$ and a simulator $\mathcal{S}$ s.t. for all environments $\mathcal{Z}_{\leq t}$ that maliciously corrupt up to t parties it holds that

$$\left\{\mathsf{REAL}_{\pi^{\mathcal{F}^{\beta-1}, \mathcal{F}_{\mathsf{BC}}^n}, \mathcal{A}_{\mathcal{D}}}\left(\mathcal{Z}_{\leq t(\lambda)}\left(1^\lambda, \cdot\right)\right)\right\}_{\lambda \in \mathbb{N}} \approx \left\{\mathsf{IDEAL}_{\mathcal{F}^{n(\lambda)}, \mathcal{S}}\left(\mathcal{Z}_{\leq t(\lambda)}\left(1^\lambda, \cdot\right)\right)\right\}_{\lambda \in \mathbb{N}}.$$

We note that each following environment corrupts either $N_1 \uplus I$ or $I \uplus N_n$, i.e., at most t parties.

$\mathcal{H}_1$ Consider an environment $\mathcal{Z}_{\leq t}^0$ that initially corrupts $I \uplus N_n$, and let's the (simulated) adversary abort all setups of size at most $\beta - 1$ that contain both a neighbor of the sender S and a neighbor of the receiver R according to Corollary 1. The honest sender S gets a random message $m \leftarrow \{0,1\}^\lambda$ as input. The malicious parties $I \uplus N_n$ are simulated *honestly with faux aborts*. Concretely, the environment $\mathcal{Z}_{\leq t}^0$ modifies the internal simulation of the malicious parties as follows: When a setup $\mathcal{F}^{\beta-1}$ on parties $P' \subset P$ is aborted with $(\texttt{abort}, C)$, then the environment $\mathcal{Z}_{\leq t}^0$ provides the simulated parties with a counterfeit abort message $(\texttt{abort}, P' \setminus C)$. Note that the faux aborts lead to the same Trust Graph for the honest parties N_1 and the malicious parties $I \uplus N_n$. Yet, the TG is connected, thus the protocol cannot abort, as guaranteed by Corollaries 1 and 5. Therefore, in the ideal execution,

[21] We omit setups of cardinality smaller than $\beta - 1$, this does not affect the validity of the proof.

after opening, the environment in the name of the malicious receiver learns the opened message m' from the ideal $\mathcal{F}^n_{\mathsf{COM},1:1}$. The environment decides `ideal` iff $m = m'$, and `real` otherwise.

$\mathcal{H}_2$ Here, we switch to the real execution with the same environment $\mathcal{Z}^0_{\leq t}$ and the dummy adversary. This is justified by the security of the assumed protocol $\pi^{\mathcal{F}^{\beta-1}, \mathcal{F}^n_{\mathsf{BC}}}$.

$\mathcal{H}_3$ Here, we consider a slightly different environment $\mathcal{Z}^1_{\leq t}$ which behaves the same way as $\mathcal{Z}^0_{\leq t}$ but corrupts $N_1 \uplus I$ instead of $I \uplus N_n$. Now, the malicious parties $N_1 \uplus I$ are provided with faux aborts analogously to the first hybrid. Still the sender S has the input message $m \leftarrow \{0,1\}^\lambda$, and the environment $\mathcal{Z}^1_{\leq t}$ decides `ideal` iff $m = m'$, and `real` otherwise. However, now m' is the honest receiver's output instead of the adversary's output. The output distributions of the environments $\mathcal{Z}^0_{\leq t}$ and $\mathcal{Z}^1_{\leq t}$ are identical because all parties have the same view of the TG and behave exactly the same way in both games, i.e., they behave honestly relative to their view of the TG.

$\mathcal{H}_4$ Here, we consider a slightly different environment $\mathcal{Z}^2_{\leq t}$ which behaves the same way as $\mathcal{Z}^1_{\leq t}$ but $\mathcal{Z}^2_{\leq t}$ internally simulates all setups in which only parties from $P' \subseteq N_1 \uplus I$ participate. All messages to such setups that $\mathcal{Z}^1_{\leq t}$ gives to the adversary in the name of malicious parties are suppressed, and all messages to malicious parties from such setups are replaced with the messages from the internal simulation. (From the view of the simulated parties the setups work normally.) In other words, all inputs to setups that only concern malicious parties are *sanitized*. Technically, the purpose of this step is to prevent the simulator from learning any communication between malicious parties via setups. Intuitively, the environment controls all malicious parties anyway, so malicious parties need not communicate via setups in the first place. The output distributions of the environments $\mathcal{Z}^1_{\leq t}$ and $\mathcal{Z}^2_{\leq t}$ are identical because the honest parties' view is identical to the previous hybrid, and hence the adversary's output distribution of the opened message m' is the same.

$\mathcal{H}_5$ Here, we switch back to the ideal execution with the same environment $\mathcal{Z}^2_{\leq t}$ and the simulator. Recall that the parties $N_1 \uplus I$ are malicious. If the environment's output is to be indistinguishable from the previous hybrid—as is dictated by the security of our assumed protocol, the simulator needs to extract the (malicious sender's) message m *during the commitment phase* from the malicious parties' messages to setups, i.e., from $\mathrm{ext}^{\mathtt{commit}}_{N_1 \uplus I}$. In other words, there exists an extractor $\mathcal{E}$ that on input $(\mathrm{ext}^{\mathtt{commit}}_{N_1 \uplus I}, B^{\mathtt{commit}})$ outputs m with some overwhelming probability $1 - \mu$.

Claim 1. *In $\mathcal{H}_4$ and $\mathcal{H}_5$ it holds that $\mathrm{ext}^{\mathtt{commit}}_{N_1} = \emptyset$, thus $\mathrm{ext}^{\mathtt{commit}}_{N_1 \uplus I} = \mathrm{ext}^{\mathtt{commit}}_{I}$.*

Proof. Recall that all messages from malicious parties to setups exclusively among malicious parties are suppressed. Furthermore, recall the structure of the TG after aborts according to Corollary 1 (compare Fig. 1). The crucial point is that the parties N_1 only participate in non-aborted setups (excluding broadcast) that contain exclusively malicious parties, i.e., within $N_1 \uplus I$. Thus all their setup

inputs are sanitized.
For contradiction, suppose there was a setup of cardinality $\beta - 1$ on some parties $P' \subset P$ s.t. it contains some party from N_1 and some honest party from $P \setminus (N_1 \uplus I) = N_n$, i.e.,

$$P' \cap N_1 \neq \emptyset \wedge P' \cap N_n \neq \emptyset \wedge \left|P'\right| < \beta \ . \tag{35}$$

Exactly this kind of setup is aborted according to Corollary 1. Consequently, only (some) messages from parties I are not sanitized. ■

Now, consider a final environment $\mathcal{Z}^3_{\leq t}$ that corrupts $I \uplus N_n$ (the sender is honest) but uses the extractor $\mathcal{E}$ during the commitment phase to obtain some message m' from the messages $(\text{ext}_I^{\texttt{commit}}, B^{\texttt{commit}})$. Specifically, $\mathcal{Z}^3_{\leq t}$ activates parties until all (honest) parties output (`receipt commit`), thus learning $(\text{ext}_I^{\texttt{commit}}, B^{\texttt{commit}})$. Finally, $\mathcal{Z}^3_{\leq t}$ uses the extractor $\mathcal{E}$ on input $(\text{ext}_I^{\texttt{commit}}, B^{\texttt{commit}}) \overset{\text{Claim 1}}{=} (\text{ext}_{N_1 \uplus I}^{\texttt{commit}}, B^{\texttt{commit}})$ to extract message m'. The environment $\mathcal{Z}^3_{\leq t}$ outputs `real` iff $m = m'$ where $m \leftarrow \{0,1\}^\lambda$ is the honest sender's random input message, and `ideal` otherwise. In the real execution, $\mathcal{Z}^3_{\leq t}$ will output `real` with overwhelming probability $1 - \mu$ due to the correctness of the extractor $\mathcal{E}$. In the ideal execution, the simulator has no information about m during the commitment phase (before round $\eta_{\texttt{commit}}$), thus the probability for $m = m'$ is $2^{-\lambda}$, hence $\mathcal{Z}^3_{\leq t}$ outputs `ideal` with probability $1 - 2^{-\lambda}$. The environment's probability of success is

$$2\left|\Pr\left[\mathcal{Z}^3_{\leq t} \text{ correct}\right] - 1/2\right| \geq 1 - 2^{-\lambda} - \mu \in \text{owhl}(\lambda) \tag{36}$$

where

$$\begin{aligned}\Pr\left[\mathcal{Z}^3_{\leq t} \text{ correct}\right] &= \Pr[\texttt{real}]\Pr\left[\mathcal{Z}^3_{\leq t} \to \texttt{real} \;\middle|\; \texttt{real}\right] \\ &\quad + \Pr[\texttt{ideal}]\Pr\left[\mathcal{Z}^3_{\leq t} \to \texttt{ideal} \;\middle|\; \texttt{ideal}\right] \\ &= \left(\Pr\left[\mathcal{Z}^3_{\leq t} \to \texttt{real} \;\middle|\; \texttt{real}\right] + \Pr\left[\mathcal{Z}^3_{\leq t} \to \texttt{ideal} \;\middle|\; \texttt{ideal}\right]\right)/2\end{aligned} \tag{37}$$

where the randomness is over the honest parties' and the environment's random coins (including the sender's message). This contradicts the security of the assumed protocol $\pi^{\mathcal{F}^{\beta-1}, \mathcal{F}^n_{\mathsf{BC}}}$. □

D.2 Constructions / Upper Bound

In this section we formally prove Lemma 3 and present our two constructions in detail. Obviously, for $t = n - 1$ we get $\beta = n$ which—in light of the universal Correlated-Randomness setup [30]—renders any other construction redundant, therefore we consider $t \leq n-2 \implies \beta = \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$ henceforth.

Lemma 3 (Connected graph $\implies$ few disconnected β-subgraphs). *Let $n, t \in \mathbb{N}$ s.t. $n/2 \leq t \leq n-2$, and let $\beta := \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$. Let V be a set of n vertices and let $v, v' \in V : v \neq v'$ be two different vertices. Moreover, let $E \subseteq \binom{V}{\leq 2}$ be a set of edges s.t. $G := (V, E)$ is an undirected, reflexive graph, and let $N_u := \mathrm{N}_G(u)$ be the set of neighbors of any vertex u, let*

$$M := \{V' \subseteq V \mid \quad V' \cap N_v \neq \emptyset \quad \wedge \quad V' \cap N_{v'} \neq \emptyset \quad \wedge \quad |V'| = \beta\}$$

be the set of relevant subsets of vertices that contain both a neighbor of v and a neighbor of v', and let

$$E^* \coloneqq \{\{u, u'\} \in E \mid |N_u \cap N_{u'}| \geq n - t\}$$

be the set of postprocessed[22] *edges. If for all $V' \in M$ the subgraph $G' \coloneqq (V', E \cap \binom{V'}{\leq 2})$ is disconnected, then $G^* \coloneqq (V, E^*)$ is disconnected. Furthermore, the map $\phi : G \mapsto G^*$ is efficiently computable.*

Proof. Similar to the proof of Lemma 2 this proof quite technical, so we want to give a brief intuition first. For contradiction we assume that all *relevant* subgraphs are disconnected, yet the overall graph is connected. Then there must exist some path between any two vertices. Moreover, to due to the *postprocessing* ϕ of the graph, the vertices along such a path must have some auxiliary neighbors that are not in the path themselves (the legs of the caterpillar graph in the proof of Lemma 2). We reach a contradiction by counting all vertices along any such path plus their auxiliary neighbors. It turns out that strictly more than n vertices are necessary to construct such an postprocessed, *connected* graph. Again, we recommend to keep Fig. 1 in mind as a visual guide.

First, we note that the map ϕ is computable in time $\mathcal{O}(n^3)$ by computing $|N_u \cap N_{u'}|$ for each pair $u, u' \in V$ and removing the appropriate edges $\{u, u'\}$ from E.
Let $h \coloneqq n - t$ and let $N^*_u \coloneqq \mathrm{N}_{G^*}(u)$. For contradiction we make the following assumptions

1. for each $V' \in M$ the subgraph $G' = (V', E \cap \binom{V'}{\leq 2})$ is disconnected,
2. $\forall \{u, u'\} \in E^* : |N^*_u \cap N^*_{u'}| \geq h$, yet
3. G^* is connected. (counter assumption)

Because of Assumption 3 there must be a path in G^* from any vertex $u \in N^*_v$ to any vertex $u' \in N^*_{v'}$. Consider any shortest path $W \coloneqq \{w_1 = u, ..., w_{\beta'} = u'\}$ from u to u' with length β'. Note that $W \cap N^*_v = \{u\} \wedge W \cap N^*_{v'} = \{u'\}$ and

$$\forall i \in [1, \beta' - 3] : N^*_{w_i} \cap N^*_{w_{i+3}} = \emptyset \tag{38}$$

because otherwise a shorter path from N^*_v to $N^*_{v'}$ existed. Any such path W must have length $\beta' > \beta$. Otherwise, because of Assumption 3, there existed some V' s.t. $W \subseteq V' \in M$ yet V' is connected which contradicts Assumption 1. For each two consecutive vertices w_i, w_{i+1} on the path W we define joint neighbors $J_i \coloneqq (N^*_{w_i} \cap N^*_{w_{i+1}}) \setminus W$ (excluding parties on the path itself) and notice that

$$J_i \cap J_{i+2} = \Big(N^*_{w_i} \cap N^*_{w_{i+1}} \cap N^*_{w_{i+2}} \cap N^*_{w_{i+3}}\Big) \setminus W \subseteq \Big(N^*_{w_i} \cap N^*_{w_{i+3}}\Big) \setminus W = \emptyset \tag{39}$$

[22] The postprocessing ϕ corresponds to the (repeated) application of Rule 4 of Definition 8, i.e., removing edges from parties with strictly more than t conflicts.

and $N^*_{w_i} \cap N^*_{w_{i+1}} \cap W = \{w_i, w_{i+1}\}$ due to Eq. (38). From Assumption 2, i.e., $\forall\{u,u'\} \in E^* : |N^*_u \cap N^*_{u'}| \geq h$, it follows that

$$|J_i| = \underbrace{\left|N^*_{w_i} \cap N^*_{w_{i+1}}\right|}_{\geq h} - \underbrace{|\{w_i, w_{i+1}\}|}_{=2} \geq h - 2 \ . \tag{40}$$

Note that $\forall i \in [2, \beta' - 1] : N^*_{w_i} \cap (N^*_v \mathbin{\dot\cup} N^*_{v'}) = \emptyset$, otherwise there existed some i s.t. $W' := \{w_i, ..., w_{\beta'}\}$ or $W'' := \{w_1, ..., w_i\}$ would be a shorter path from N^*_v to $N^*_{v'}$ in G^*. Moreover

$$\forall i \in [2, \beta' - 2] : J_i \cap (N^*_v \mathbin{\dot\cup} N^*_{v'}) = \emptyset \tag{41}$$

implies

$$\forall i \in [1, \lfloor \beta'/2 \rfloor - 1] : J_{2i} \cap (N^*_v \mathbin{\dot\cup} N^*_{v'}) = \emptyset \ . \tag{42}$$

Now we can express the set of all vertices as disjoint sets

$$V \supseteq W \mathbin{\dot\cup} (N^*_v \setminus W) \mathbin{\dot\cup} (N^*_{v'} \setminus W) \mathbin{\dot\cup} \dot{\bigcup_{i=1}^{\lfloor \beta'/2 \rfloor - 1}} J_{2i} \tag{43}$$

which implies

$$\begin{aligned} n = |V| &\geq \underbrace{|W|}_{\beta'} + \underbrace{|N^*_v \setminus W|}_{\geq h-1} + \underbrace{|N^*_{v'} \setminus W|}_{\geq h-1} + \sum_{i=1}^{\lfloor \beta'/2 \rfloor - 1} \underbrace{|J_{2i}|}_{\geq h-2} \\ &\geq \beta' + 2(h-1) + \lfloor \beta'/2 - 1 \rfloor (h-2) \\ &= \beta' + 2 + \lfloor \beta'/2 + 1 \rfloor (h-2) \\ &\geq \beta + 3 + \lfloor (\beta+3)/2 \rfloor (h-2) \ . \end{aligned} \tag{44}$$

For $n/h \in \mathbb{N} \iff \beta = 2n/h - 2$ we get

$$n \geq 2n/h + 1 + (n/h)(h-2) = n + 1 \ . \tag{45}$$

For $n/h \notin \mathbb{N} \iff \beta = 2\lfloor n/h \rfloor - 1$ we get

$$n \geq 2\lfloor n/h \rfloor + 2 + (\lfloor n/h \rfloor + 1)(h-2) = \lfloor n/h \rfloor h + h = \lceil n/h \rceil h > n \ . \tag{46}$$

Both cases come to a contradiction which concludes the proof. Note that Eq. (45) poses a "minimal" contradiction, in the sense that the overall number of vertices exceeds n by just one. This attests that the setup size β is indeed optimal, matching the tightly complementing Lemma 2. $\square$

The above lemma can be understood visually when looking at Fig. 1. There is always a connected subgraph of G of size β which connects some neighbor (in grey) of the first vertex and some neighbor of the last vertex.

Next, we give a construction of a global (one-to-many) commitment functionality from commitment setups of size β and a global broadcast. The basic idea is for the sender to simply input its message into all setups, and later open all of them. If the sender is honest, all messages will be consistent and all honest parties will output the correct message. A problem arises when a) the sender is malicious and inputs different messages into different setups, or b) many setups are aborted s.t. some (honest) receivers don't obtain any opening information. For a) the remedy is to let the sender commit to a threshold sharing of its message such that the receivers can request the opening of some shares—which the sender then also has to broadcast—because the threshold sharing is robust against a half of manipulated shares any sufficient inconsistency between the shares obtained from the setup commitment and the ones broadcasted will be detected with overwhelming probability. For b) we use the structure of the TG to recognize that whenever a receiver was completely "cut off", most of its neighbors must be honest, thus such receivers simply follow the behavior of their neighbors.

Theorem 3 (COM expansion). *Let $n = n(\lambda), t = t(\lambda), \beta := \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$ s.t. $n/2 \leq t \leq n-2$ and $\binom{n}{\beta} \in \mathrm{poly}(\lambda)$. There is an efficient protocol π_{COM} that statistically securely UC-realizes $\mathcal{F}^n_{\mathsf{COM}}$ in the $\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against environments that (maliciously) corrupt up to t parties. Formally,*

$$\left\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta, \mathcal{F}^n_{\mathsf{BC}}\right\} \overset{\text{stat}}{\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{COM}} \ . \tag{2}$$

Proof. First, note that $\mathcal{F}^n_{\mathsf{TG}}$ is realizable using only $\mathcal{F}^n_{\mathsf{BC}}$ by Lemma 6, so we explicitly use $\mathcal{F}^n_{\mathsf{TG}}$ in the protocol construction. We require all honest parties to locally compute the postprocessed Trust Graph $G^* \leftarrow \phi(G)$ when obtaining G from $\mathcal{F}^n_{\mathsf{TG}}$. Moreover, we require π_{COM} to be abort-respecting as described in Definition 8. In particular, whenever a setup is aborted, the honest parties resolve that abort according to Rule 2 before resuming the protocol s.t. the corresponding subTG is disconnected.
Before we give a formal description we outline the protocol intuitively. Let $P = \{\mathsf{S}, \mathsf{R}_1, ..., \mathsf{R}_{n-1}\}$ denote the set of parties where S is the sender and R_i are the receivers. For notational purposes we use the notation $\mathsf{R}_0 := \mathsf{S}$. Denote by $N_{\mathsf{P}} := \mathrm{N}_G(\mathsf{P})$ the party P's neighbors at any given point in time. Let $\ell := n\lambda^2$ be the size of the $(\ell, 2\ell)$-threshold sharing and let $\rho := \lambda$ be the number of probing shares per party.[23] The commit phase goes as follows:

1. The sender draws a random bits $b \leftarrow \{0,1\}^n$ and secret-shares its message m as $\mu \leftarrow \mathrm{Share}_{\ell,2\ell}(m)$ and each bit of b as $\xi^j \leftarrow \mathrm{Share}_{\ell,2\ell}(b_j)$. Then the sender broadcasts $(\texttt{commit}, \sigma, N_{\mathsf{S}})$ where $\sigma := \mu \oplus \bigoplus_{\mathsf{R}_j \in N_{\mathsf{S}}} \xi^j$ is a masked sharing, and sends ξ^j to the neighboring receiver R_j. Additionally, the sender inputs the sharing μ into all non-aborted SCOM setups.
2. Each receiver R_j who obtained some sharing ξ^j inputs it into all non-aborted SCOM setups and acknowledges with a broadcasted receipt.
3. Each party draws random probing indices $\nu^j \leftarrow \binom{[2\ell]}{\rho}$ and broadcasts them.
4. Each party opens all its SCOMs on all probed indices $U := \bigcup_{j=0}^{n-1} \nu^j$ and additionally broadcasts its respective shares, i.e., μ^{BC}_U or $\xi^{\mathrm{BC},j}_U$.
5. Each party checks the consistency between the probed shares and the broadcasted one. If the shares opened in a setup are not equal to the broadcasted

[23] We use these (non-optimal) parameters for simple exposition.

ones, then the parties that notice the inconsistency proceed to declare conflicts with the parties whose broadcasted shares are not equal to the ones opened in the SCOM. Consequently, such setups will become aborted (disconnected) after the application of Rule 2, because honest parties declare conflicts with the offender and its loyalists. Next, the parties check consistency of the broadcasted shares themselves, i.e., $\sigma_U \overset{?}{=} \mu_U^{\mathrm{BC}} \oplus \bigoplus_{\mathsf{R}_j \in N_\mathsf{S}} \xi_U^{\mathrm{BC},j}$.

- If all shares are consistent, each receiver outputs a receipt.
- Otherwise, the sender must identify which neighbors broadcasted a different $\xi^{\mathrm{BC},j} \neq \xi^j$. As such at least one sender–neighbor conflict arises, the protocol starts again with updated neighbors N_S. If no such conflict is declared, then S must be malicious.

The opening phase goes as follows:

1. Each party opens all SCOM setups in which it committed to some shares.[24]
2. Each receiver that receives all necessary shares reconstructs the message m and broadcasts it.
3. Each receiver declares a conflict with each receiver that broadcasts another message, and outputs m.
4. Each receiver who did not receive all sufficient shares either abort the protocol if the TG is disconnected or they output the majority of what their neighbors broadcasted.

The crux of the protocol can be made intuitive by looking at Fig. 1c). Suppose the malicious sender is represented as 1 and the honest parties are $7, 8, 9$. Now, the adversary can abort setups in such a way that the sender and honest parties are not jointly in any setup—thus cutting off the honest parties 8 and 9 from obtaining any opening information. However, the party 7 will always be in some setup with each neighbor, 2 and 3, of the sender. Recall that the sender gives its neighbors an additive share ξ^j of the mask. Because the sender's neighbors are committed towards party 7, party 7 can recover the message and parties 8 and 9 can output the same message because they know that 7 is honest as well. On the other hand, if the sender was honest, then malicious receivers don't obtain any information about the message from σ and the ξ^j's because at least one additive share of the mask is held by an honest neighbor of the sender.

Now, we provide a more formal description of the protocol. Throughout the protocol let $\mathfrak{I}(S) \coloneqq \{M \subseteq P \mid |M| = \beta \wedge S \subseteq M \wedge (M, E^* \cap \binom{M}{\leq 2}) \text{ connected}\}$ be the set of subsets of parties containing the parties S whose corresponding subgraph is (still) connected, i.e., whose corresponding setup is not (yet) aborted. In other words, $\mathfrak{I}(S)$ is the set of subsets of parties with which the parties S can still perform a setup. Note that this set might dynamically change during the protocol as setups are aborted. We use the number $\iota \coloneqq |N_\mathsf{S}|$ as a counter, i.e., everytime ι decreases the protocol restarts—all honest parties deprecate all previously used information and setups, and the sender behaves as if it received $(\texttt{input}, m)$. We denote an instance of $\mathcal{F}_{\mathsf{SCOM}}^{\beta}$ on some set of parties $M \in \mathfrak{I}(\emptyset)$

[24] Due to the verification step in the commit phase the shares in each setup must reconstruct to the same message except with negligible probability.

with counter ι by $\mathcal{C}_{M,\iota}$. As a notational guide we use the index j for receivers that are neighbors with S while the index $i \in [0, n-1]$ is used for any receiver.

Protocol π_{COM}

The protocol π_{COM} proceeds as follows, running with parties $P = \{\mathsf{S}, \mathsf{R}_1, ..., \mathsf{R}_{n-1}\}$, malicious parties $C \subseteq P$, adversary $\mathcal{A}$ and environment $\mathcal{Z}$. Messages not covered here are ignored.

- **In the first round** each party $G^*_{\text{ref}} \leftarrow (P, \binom{P}{\leq 2})$ and stores $\iota_{\text{ref}} \leftarrow \iota$.
- **Each round** each party P inputs $(\texttt{query})$ into $\mathcal{F}^n_{\mathsf{TG}}$ to obtain the TG in the next round.
- **On output** $G = (P, E)$ from $\mathcal{F}^n_{\mathsf{TG}}$, the party P computes the current counter ι. If $\iota < \iota_{\text{ref}}$, it deletes all previously received messages and updates the stored $G^*_{\text{ref}} \leftarrow \phi(G)$ and sets $\iota_{\text{ref}} \leftarrow \iota$. If the sender received $(\texttt{commit}, m)$ from $\mathcal{Z}$ before, it behaves as if it received it again.
- **On input** $(\texttt{commit}, m \in \{0,1\})$ from $\mathcal{Z}$, the sender S creates a sharing $\mu \leftarrow \text{Share}_{\ell,2\ell}(m)$, and inputs $(\texttt{commit}, \mu)$ into $\mathcal{C}_{M,\iota}$ for each $M \in \mathfrak{I}(\{\mathsf{S}\})$. The sender draws bits $b \leftarrow \{0,1\}^{n-1}$, shares each bit $\xi^j \leftarrow \text{Share}_{\ell,2\ell}(b_j)$ and sends $(\texttt{shares}, \xi^j)$ to R_j via $\mathcal{F}^2_{\mathsf{SMT}}$. Then S broadcasts $(\texttt{commit}, \sigma, N_{\mathsf{S}})$ where $\sigma := \mu \oplus \bigoplus_{\mathsf{R}_i \in N_{\mathsf{S}}} \xi^i$. The sender ignores further input of this type except when the counter ι is decreased.
- **On output** $(\texttt{shares}, \xi^j)$ from $\mathcal{F}^2_{\mathsf{SMT}}$ from S, the party R_j stores ξ^j and inputs $(\texttt{commit}, \xi^j)$ into all $\mathcal{C}_{M,\iota}$ for each $M \in \mathfrak{I}(\{\mathsf{R}_j\})$. The receiver R_j broadcasts $(\texttt{receipt})$.
- **On output** $(\texttt{output}, \mathsf{S}, (\texttt{commit}, \sigma, N'))$ from $\mathcal{F}^n_{\mathsf{BC}}$, the receiver R_i checks whether $N' = N_{\mathsf{S}}$, if not the message is ignored. Any receiver $\mathsf{R}_j \in N'$ also checks whether it received $(\texttt{shares}, \xi^j)$ before. If not, then it inputs $(\texttt{conflict}, \mathsf{S})$ into $\mathcal{F}^n_{\mathsf{TG}}$. (Note that ι decreases by one.) Next, R_i checks if it received $(\texttt{receipt}, \mathsf{R}_j)$ from $\mathcal{C}_{M,\iota}$ for all $\mathsf{R}_j \in N_{\mathsf{S}}$ and $M \in \mathfrak{I}(\{\mathsf{R}_i, \mathsf{R}_j\})$. If so, R_i samples some probing indices $\nu^i \leftarrow \binom{[2\ell]}{\rho}$ and broadcasts $(\texttt{probe}, \nu^i)$. Then R_i stores $N_{\text{fix}} \leftarrow N'$.
- **On output** $(\texttt{output}, \mathsf{R}_{i'}, (\texttt{probe}, \nu^{i'}))$ from $\mathcal{F}^n_{\mathsf{BC}}$ for all $i' \in [0, n-1]$, each party $\mathsf{R}_i : i \in [0, n-1]$ opens its shares at the probed indices by inputting $(\texttt{open}, U)$ into $\mathcal{C}_{M,\iota}$ for each $M \in \mathfrak{I}(\{\mathsf{R}_i\})$ where $U := \bigcup_{i'=0}^{n-1} \nu^{i'}$. Furthermore, each neighbor receiver $\mathsf{R}_j \in N_{\text{fix}}$ broadcasts the probed shares $(\texttt{shares}, \xi^j_U)$ while S broadcasts $(\texttt{shares}, \mu_U)$.
- **On output** $(\texttt{output}, \mathsf{S}, U, \mu_U)$ from $\mathcal{C}_{M,\iota}$, the receiver R_i stores $\mu^M_U \leftarrow \mu_U$.
- **On output** $(\texttt{output}, \mathsf{R}_j, U, \xi^j_U)$ from $\mathcal{C}_{M,\iota}$, the party R_i stores $\xi^{j,M}_U \leftarrow \xi^j_U$.
- **On output** $(\texttt{output}, \mathsf{S}, (\texttt{shares}, \mu^{\text{BC}}_U))$ from $\mathcal{F}^n_{\mathsf{BC}}$, the receiver R_i checks consistency with the opened shares. If $\exists M \in \mathfrak{I}(\{\mathsf{S}, \mathsf{R}_i\}) : \mu^M_U \neq \mu^{\text{BC}}_U$, then receiver R_i declares a conflict with S, i.e., it inputs $(\texttt{conflict}, \mathsf{S})$ into $\mathcal{F}^n_{\mathsf{TG}}$. In effect, the setup $\mathcal{C}_{M,\iota}$ is aborted with $(\texttt{abort}, \mathsf{S})$, hence Rule 2 of Definition 8 applies and the subgraph on M becomes disconnected. (Note that ι decreases.)

Protocol π_{COM} (cont'd)

- **On output** $(\texttt{output}, \mathsf{R}_j, (\texttt{shares}, \xi_U^{\mathrm{BC},j}))$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $\mathsf{R}_j \in N'$, the receiver R_i checks consistency with the opened shares. If $\exists M \in \mathfrak{I}(\{\mathsf{R}_i, \mathsf{R}_j\}) : \xi_U^{M,j} \neq \xi_U^{\mathrm{BC},j}$, then receiver R_i declares a conflict with R_j, i.e., it inputs $(\texttt{conflict}, \mathsf{R}_j)$ into $\mathcal{F}_{\mathsf{TG}}^n$. In effect, the setup $\mathcal{C}_{M,\iota}$ is aborted with $(\texttt{abort}, \mathsf{R}_j)$, hence Rule 2 of Definition 8 applies and the subgraph on M becomes disconnected. For each $\mathsf{R}_j \in N'$ the sender checks whether $\xi_U^{\mathrm{BC},j} = \xi_U^j$, if not, then S inputs $(\texttt{conflict}, \mathsf{R}_j)$ into $\mathcal{F}_{\mathsf{TG}}^n$. (Note that ι decreases.) Furthermore, if $\sigma_U \neq \mu_U^{\mathrm{BC}} \oplus \bigoplus_{\mathsf{R}_j \in N_{\mathrm{fix}}} \xi_U^{\mathrm{BC},j}$, then receiver R_i stores an inconsistency flag $\delta \leftarrow 1$, otherwise $\delta \leftarrow 0$. Each party broadcasts $\texttt{OK}^1$.
- **On output** $(\texttt{output}, \mathsf{R}_{i'}, \texttt{OK}^1)$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $i' \in [0, n-1]$, then
  - R_i aborts[a] with $(\texttt{abort}, \mathsf{S})$, if $\delta = 1 \wedge N_{\mathsf{S}} = N_{\mathrm{fix}}$,
  - otherwise R_i broadcasts $\texttt{OK}^2$, if $\delta = 0 \wedge N_{\mathsf{S}} = N_{\mathrm{fix}}$.
- **On output** $(\texttt{output}, \mathsf{R}_i, \texttt{OK}^2)$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $i \in [0, n-1]$, each party outputs $(\texttt{receipt commit})$ and stores $\iota_{\mathrm{fix}} \leftarrow \iota$.
- **On input** $(\texttt{open})$ from $\mathcal{Z}$, sender S opens all shares by inputting $(\texttt{open}, [2\ell])$ into $\mathcal{C}_{M,\iota_{\mathrm{fix}}}$ for each $M \in \mathfrak{I}(\{\mathsf{S}\})$, and broadcasts $\texttt{open}$.
- **On output** $(\texttt{output}, \mathsf{S}, [2\ell], \mu_{[2\ell]} = \mu)$ from $\mathcal{C}_{M,\iota_{\mathrm{fix}}}$, the receiver R_i stores μ.
- **On output** $(\texttt{output}, \mathsf{S}, \texttt{open})$ from $\mathcal{F}_{\mathsf{BC}}^n$, any receiver $\mathsf{R}_j \in N_{\mathrm{fix}}$ opens its SCOMs by inputting $(\texttt{open}, [2\ell])$ into $\mathcal{C}_{M,\iota_{\mathrm{fix}}}$. If receiver R_i obtained μ, i.e., $\mathfrak{I}(\{\mathsf{S}, \mathsf{R}_i\}) \neq \emptyset$, then R_i broadcasts and stores message $m \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu)$.
- **On output** $(\texttt{open}, \mathsf{R}_j, \xi_{[2\ell]}^j = \xi^j)$ from $\mathcal{C}_{M,\iota_{\mathrm{fix}}}$ for all $\mathsf{R}_j \in N_{\mathrm{fix}}$, then R_i recovers $m \leftarrow \mathrm{Recover}_{\ell,2\ell}(\sigma \oplus \bigoplus_{\mathsf{R}_j \in N_{\mathrm{fix}}} \xi^j)$ and broadcasts and stores m.
- **On output** $(\texttt{output}, \mathsf{S}, \texttt{open})$ and $(\texttt{output}, \mathsf{R}_{i'}, \widetilde{m}^{i'})$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $\mathsf{R}_{i'} \in \mathrm{N}_G(\mathsf{R}_i)$, the receiver R_i inputs $(\texttt{conflict}, \mathsf{R}_{i'})$ into $\mathcal{F}_{\mathsf{TG}}^n$ for each $\widetilde{m}^{i'} \neq m$, i.e., R_i declares a conflict with each receiver broadcasting a message different from its own. Afterwards, Rule 2 of Definition 8 is applied. If the TG is disconnected, then R_i aborts, otherwise R_i outputs $(\texttt{open}, m)$ and terminates. Either R_i recovered m itself, or it outputs the majority bit of whatever its neighbors broadcasted.

[a] Some receiver broadcasted inconsistent shares but the sender did not declare a conflict with them.

Before analyzing the protocol in detail we give a simulator for the canonical dummy adversary. The simulator runs a simulated protocol where it executes the protocol code for all non-corrupted parties. Whenever the environment activates an honest dummy party, the simulator is notified and simulates the party's protocol code. Any message from and to malicious parties are forwarded to the simulated setup functionalities.

Simulator for π_{COM}

- **On output** (`receipt commit`) from $\mathcal{F}^n_{\mathsf{COM}}$, the simulator $\mathcal{S}$ gives input $(\texttt{commit}, 0)$ to the simulated sender S'.
- **On output** (`receipt commit`) from all simulated (honest) R'_i, the simulator $\mathcal{S}$ sends $(\texttt{commit}, \widetilde{m})$ for $\widetilde{m} \leftarrow \text{Recover}_{\ell,2\ell}(\widetilde{\mu})$ to the ideal $\mathcal{F}^n_{\mathsf{COM}}$ in the name of S which previously input the sharing $\widetilde{\mu}$ into the setup $\mathcal{C}_{M',\iota_{\text{fix}}}$ s.t. M' is the canonically smallest set in $\mathfrak{I}(\{\mathsf{S}\})$.
- **On output** $(\texttt{open}, m)$ from $\mathcal{F}^n_{\mathsf{COM}}$, the simulator $\mathcal{S}$ gives local input (`open`) to the simulated sender S' which will open the remaining setups $\mathcal{C}_{M,\iota_{\text{fix}}}$ for all $M \in \mathfrak{I}(\{\mathsf{S}\})$. Here, the simulator equivocates each remaining simulated $\mathcal{C}_{M,\iota_{\text{fix}}}$ to a random sharing $\widetilde{\mu} \leftarrow \text{Share}_{\ell,2\ell}(m)$ s.t. $\widetilde{\mu}_U = \mu_U$. The simulated $\mathcal{C}_{M,\iota_{\text{fix}}}$ on input $(\texttt{open}, [2\ell])$ from S' sends $(\texttt{open}, \mathsf{S}, [2\ell], \widetilde{\mu})$ to the simulated dummy adversary. This is possible because at most $n\rho$ shares have been probed (at least $2\ell - n\rho$ remain veiled) but the simulator only needs to equivocate $\ell + 1 = n\lambda^2 + 1 \leq n(2\lambda^2 - \lambda) = 2\ell - n\rho$. The same strategy applies for the opening of the ξ^j's of the honest receivers.
- **On output** $(\texttt{open}, m)$ from all simulated R'_j, the simulator $\mathcal{S}$ sends (`open`) to $\mathcal{F}^n_{\mathsf{COM}}$ in the name of S.
- **On input** $(\texttt{abort}, C')$ for any simulated setup session with parties $M \subseteq P$ and $\emptyset \neq C' \subset M \cap C$, the simulator aborts that session by forwarding $(\texttt{abort}, C')$ to it.
- **On output** $(\texttt{abort}, C')$ from all simulated parties, the simulator $\mathcal{S}$ inputs $(\texttt{abort}, C')$ into $\mathcal{F}^n_{\mathsf{COM}}$.

From the description of the simulator it is apparent that the honest dummy parties output the receipt resp. the opened value exactly when the simulated (honest) parties would output the receipt resp. opened value. It is also clear from the simulator's description that for an honest party the simulator can equivocate its input into simulated setup functionalities to let the simulated parties output $\widetilde{m}$ which the simulator receives from $\mathcal{F}^n_{\mathsf{COM}}$ at the start of the opening phase.

Henceforth, we focus on proving that honest (simulated) receivers actually output $\widetilde{m}$. To see why the protocol works, we make four observations conditioned on the fact that the protocol does not abort. Let $H \subseteq P$ denote the honest parties.

Claim 2. *When all (honest) parties output* (`receipt commit`) *in the real/simulated protocol, for all $M \in \mathfrak{I}(\{\mathsf{S}\})$ s.t. $M \cap H \neq \emptyset$ the sharings $\widetilde{\mu}^M$ in $\mathcal{C}_{M,\iota_{\text{fix}}}$ encode the same message $\widetilde{m}$ with overwhelming probability. The same holds for the sharings $\widetilde{\xi}^{M,j}$ for all $\mathsf{R}_j \in N_{\text{fix}}$ and $M \in \mathfrak{I}(\{\mathsf{R}_j\})$ s.t. $M \cap H \neq \emptyset$.*

Proof. We prove Claim 2 by contradiction. Note that all sets M contain at least one honest party. Therefore, when all (honest) parties output (`receipt commit`), then there exists a set of (at least ρ uniformly chosen) indices ν on which the shares $\widetilde{\mu}^M$ of all setups must equal the broadcasted shares $\widetilde{\mu}^{\text{BC}}$. Formally,

$$\exists \nu \in \binom{[2\ell]}{\rho} \forall M \in \mathfrak{I}(\{\mathsf{S}\}) : M \cap H \neq \emptyset \implies \widetilde{\mu}^M_\nu = \widetilde{\mu}^{\text{BC}}_\nu \ . \tag{47}$$

Now, suppose for contradiction that there exist two setups M and M' whose *valid* sharings encode different messages. Because the sharings are valid they must differ on at least $\ell+1$ indices, i.e., $\exists \widetilde{\nu} \in \binom{[2\ell]}{\ell+1} \forall \kappa \in \widetilde{\nu} : \widetilde{\mu}_\kappa^M \neq \widetilde{\mu}_\kappa^{M'}$, while also $\nu \cap \widetilde{\nu} = \emptyset$. By Lemma 5 the probability of that event is bounded by

$$\Pr_{\nu \leftarrow \binom{[2\ell]}{\rho}}[\widetilde{\nu} \cap \nu = \emptyset] \leq 2^{-\rho(\ell+1)/2\ell} \leq 2^{-\rho/2} = 2^{-\lambda/2} \ . \tag{48}$$

Conversely, with probability at least $1 - 2^{-\lambda/2}$ two sharings encode the same message. The same argument holds for the masks $\xi^{M,j}$. By a union bound over all possible commitments (with the maximum of n restarts) we know that all mask bit and the masked message itself are consistent with probability at least $1 - n^3 2^{-\lambda/2}$. ■

Recall that the simulator extracts the message $\widetilde{m}$ from the malicious sender's inputs into the setups during the commitment phase. Claim 2 combined with the fact that (`receipt commit`) is only output if $\sigma = \mu \oplus \bigoplus_{\mathsf{R}_i \in N_{\text{fix}}} \xi^i$ guarantees that $\widetilde{m}$ is equal to whatever message (if any) *all* honest receivers output during the opening phase. For an honest sender it obviously holds that $\widetilde{m} = m$.

Next, we show that honest receivers can indeed output the message $\widetilde{m}$. To this end we first show that *some* honest receiver can output $\widetilde{m}$ by recovering it from at least one setup.

Claim 3. *For $\mathsf{S} \notin H$, at least one honest receiver is able to reconstruct the message by receiving all shares:* $\exists \mathsf{R} \in H \, \exists \mathsf{N} \in N_\mathsf{S} : \mathfrak{I}(\{\mathsf{N}, \mathsf{R}\}) \neq \emptyset$.

Proof. Again, we prove the observation by contradiction. First, note that the map $\phi : G \mapsto G^*$ preserves the "no honest-honest-conflict" property of the TG. That is, for honest parties $H \subset P$, if $\binom{H}{\leq 2} \subseteq E \implies \binom{H}{\leq 2} \subseteq E^*$. This follows readily from the fact that honest parties are always in some (possibly many) h-clique in G whereas the postprocessing ϕ removes edges between exactly those parties that are not jointly in any h-clique in G. Therefore at least one party of any removed edge must be malicious.

If an honest receiver R is in some setup with S itself or in some setup with *each* neighbor of the sender, the receiver R can recover the correct message. In the first case R can directly recover $m \leftarrow \text{Recover}_{\ell,2\ell}(\mu)$, in the second case R can recover $m \leftarrow \text{Recover}_{\ell,2\ell}(\sigma \oplus \bigoplus_{\mathsf{R}_i \in N_{\text{fix}}} \xi^i)$ because R obtains all ξ^j's of all neighbors of the sender.

Now, suppose for contradiction $\forall \mathsf{R} \in H \forall \mathsf{N} \in N_\mathsf{S} : \mathfrak{I}(\{\mathsf{N}, \mathsf{R}\}) = \emptyset$. The statement $\mathfrak{I}(\{\mathsf{N}, \mathsf{R}\}) = \emptyset$ means that all setups on parties M s.t. $\mathsf{N}, \mathsf{R} \in M \wedge |M| = \beta$ must be aborted. To bring this counter assumption to a contradiction we use the counting argument from Lemma 3 where $V := P$, $v = \mathsf{S}$ and $v' = \mathsf{H} \in H \subset P$. Note that any $\mathsf{N_S} \in N_\mathsf{S}$ and $\mathsf{N_H} \in H \subseteq N_\mathsf{H}$ must have distance at least $\Delta_G(\mathsf{N_S}, \mathsf{N_H}) > \beta$ by assumption. We consider the two cases:

1. Some honest party has a large distance to the sender, i.e., $\exists \mathsf{H} \in H \forall \mathsf{N_S} \in N_\mathsf{S} : \Delta_G(\mathsf{N_S}, \mathsf{H}) \geq \beta + 2$. Then any $\mathsf{N_S} \in N_\mathsf{S}, \mathsf{N_H} \in N_\mathsf{H}$ have at least distance $\beta' := \beta + 1 > \beta$. Hence, counting the overall number of parties according to Eqs. (44) to (46) yields a contradiction.

2. All honest parties have small distance to the sender, i.e., $\exists \mathsf{N}_\mathsf{S} \in N_\mathsf{S} \setminus \{\mathsf{S}\}\, \forall \mathsf{H} \in H : \Delta_G(\mathsf{N}_\mathsf{S}, \mathsf{H}) = \beta + 1$. Let N'_H be some neighbor of the honest parties with distance $\Delta_G(\mathsf{S}, \mathsf{N}'_\mathsf{H}) = \beta = \Delta_G(\mathsf{N}_\mathsf{S}, \mathsf{H})$. Here, we can count the overall number of parties in L- and S-segments corresponding to a shortest path from S to N'_H. Overall, we count $n \geq \lceil(\beta+1)/2\rceil + \lfloor(\beta+1)/2\rfloor(h-1) + h$ parties, i.e., the parties in the L-segments, the parties in the S-segments, plus the honest parties. We find the contradictions
$$\begin{aligned} \beta = 2n/h - 2 &\implies n \geq (n/h - 1)(h-1) + n/h + h = n + 1 \\ \beta = 2\lfloor n/h \rfloor - 1 &\implies n \geq (\lfloor n/h \rfloor - 1)(h-1) + \lfloor n/h \rfloor + h > n \;. \end{aligned} \tag{49}$$

Consequently, if no abort occurs, at least one honest receiver must obtain $\widetilde{m}$. ■

Lastly, we show that any receiver that does not obtain $\widetilde{m}$ from some setup can rely on its neighbors to output $\widetilde{m}$.

Claim 4. *If an honest "cut-off" receiver does not receive any shares and the TG is connected, it can output whichever message the majority of its neighbors output.*

Proof. Note that for $h = 2$ a receiver that does not recover the message directly has only one neighbor who must be honest. For $h \geq 3$ at most $h-1$ neighbors of an cut-off receiver can recover the message directly. Furthermore, to the cut-off receiver R_i it is obvious from the TG which ones those are, i.e., $\mathrm{N}_G(\mathsf{R}_i) \cap \bigcup_{\mathsf{N} \in \mathrm{N}_G(\mathsf{S})} \bigcup_{M \in \mathfrak{I}(\{\mathsf{N}\})} M$. We want to recall Fig. 1b) as a visual aid. Here, suppose 1 is the sender and 8 is the cut-off receiver, then only 5 and 6 are the receivers that recover the message from the masked sharing σ and the mask sharings ξ^j obtained from 2 and 3. If 5 and 6 are honest, they both broadcast m. If one deviates and broadcasts something else, then 5 and 6 get into a conflict As a consequence of applying the map ϕ we now also find conflicts between 4 and 5, and 4 and 6, thus the overall TG is disconnected and all (honest) parties can abort.
More generally, if a receiver R is cut off, then the distance between the sender and the receiver is large. W.l.o.g. the sender is in the first segment, and the cut-off receiver is in the last segments, i.e., $\mathsf{R} \in L_{\lfloor n/h \rfloor} \uplus S_{\lceil n/h \rceil - 1}$.

- If $|L_{\lfloor n/h \rfloor} \uplus S_{\lceil n/h \rceil - 1}| = h$, then all neighbors of an honest R are honest as well because $\mathsf{R} \in L_{\lfloor n/h \rfloor} \uplus S_{\lceil n/h \rceil - 1} = H$.
- If $|L_{\lfloor n/h \rfloor} \uplus S_{\lceil n/h \rceil - 1}| < h$, then the parties in $S_{\lceil n/h \rceil - 2}$ form a majority of the parties $R' := (S_{\lceil n/h \rceil - 1} \uplus L_{\lfloor n/h \rfloor} \uplus S_{\lceil n/h \rceil - 1}) \setminus \{\mathsf{R}\} \supseteq N_\mathsf{R} \setminus \{\mathsf{R}\}$. Moreover, all parties in $S_{\lceil n/h \rceil - 2}$ must have broadcasted the same message, otherwise they would be in conflict, and hence not in the same segment. Furthermore, at least one party in $S_{\lceil n/h \rceil - 2}$ must be honest as well, because $h-1$ of the at most $2h-3$ parties in R' are honest. As such all parties in $S_{\lceil n/h \rceil - 2}$ (the majority) must broadcast the correct message.

■

For a malicious sender we have shown that the simulator extracts the message $\widetilde{m}$ that the honest receivers actually output in the opening phase. We make a last observation for the case of an honest sender.

Claim 5. *If the sender is honest and the protocol is not aborted, then all honest receiver will output the correct message.*

Proof. If the sender is honest it will truthfully input its sharing μ into $\mathcal{C}_{M,\iota_{\mathrm{fix}}}$ for all $M \in \mathfrak{I}(\{\mathsf{S}\})$. As such, if a receiver R recovers the message from some direct setup $M \in \mathfrak{I}(\{\mathsf{S}, \mathsf{R}\})$, then it obviously recovers the correct message. Here the adversary has no way of interfering with the sharing of the message. Lastly, note that if S is honest and R is honest as well, then $\mathfrak{I}(\{\mathsf{S}, \mathsf{R}\}) \neq \emptyset$ because $\{\mathsf{S}, \mathsf{R}\}$ plus their extended neighbors always form a connected subgraph of size β of the complement TG, otherwise the TG would be disconnected and the protocol aborted. ■

For exhibition, we can now give a series of hybrid games:

$\mathcal{H}_1$ This game is the real execution where all parties follow the protocol.

$\mathcal{H}_2$ This game is the same as before except that there is an ideal dummy functionality $\mathcal{F}^n_{\mathsf{COM}}$ which is connected to (honest) dummy parties which obtain the same local inputs as the honest parties in the real protocol. The real adversary $\mathcal{A}$ is replaced with a simulator $\mathcal{S}$ that behaves exactly as $\mathcal{A}$ and, if the sender is malicious, the simulator $\mathcal{S}$ extracts the sender's input $(\texttt{commit}, m)$ for $\mathcal{F}^n_{\mathsf{COM}}$ from the real protocol execution. If the sender is honest, then the honest dummy sender forwards its input $(\texttt{commit}, m)$ to $\mathcal{F}^n_{\mathsf{COM}}$.
The output of the dummy parties is ignored, i.e., the game's output is still exactly the same as before.

$\mathcal{H}_3$ This game is the same as before except that this game aborts, iff the (opened) message output by the honest parties differs from the output of the dummy parties interacting with the ideal dummy functionality.
This event of differing outputs occurs only negligibly often, as argued in Claims 2 to 5. Hence, the game's output behavior is statistically close to the previous one.

$\mathcal{H}_4$ This game is the same as before except that now the environment interacts with the the dummy parties of the ideal functionality instead of the parties in the (real) execution output. (Still the parties of the protocol execution get the same input as the dummy parties, i.e., they behave as before. The only difference is that their output is ignored.)
Due to the abort condition established in the previous game these outputs are identical, hence the output of this game is exactly the same as on the previous game.

$\mathcal{H}_5$ This game is the same as before except that the parties in the protocol no longer obtain the environment's input but are instead simulated by the simulator $\mathcal{S}$. In particular, if the simulator gets $(\texttt{receipt commit})$ from the ideal $\mathcal{F}^n_{\mathsf{COM}}$ it simulates the honest sender with input $(\texttt{commit}, 0)$. This game is the ideal execution.

This game and the previous game have exactly the same output behavior because the (honest) dummy parties' output depends solely on the ideal functionality $\mathcal{F}^n_{\mathsf{COM}}$ and not on the local input of the simulated sender.

The protocol requires invocation of at most $\mathcal{O}(\binom{n}{\beta}) \subseteq \text{poly}(\lambda)$ setups per restart (decrease of ι) and can be restarted at most $\mathcal{O}(n) \subseteq \text{poly}(\lambda)$ times because a restart means that the sender lost at least one neighbor. $\square$

Next, we present our protocol for Fully Committed Oblivious Transfer (FCOT). As in the classical OT the sender holds two messages m^0, m^1 and the receiver holds a choice bit c. To make the construction easier to analyze we assume parties to have access to the global commitment setup which is provided by Theorem 3. The role of the global commitment is to commit both the sender and the receiver to their input s.t. they cannot change it after the OT phase.

Intuitively, the protocol tries to perform a *direct* Shared Oblivious Transfer (SOT) via a setup that contains both the sender and the receiver. If this does not work, then the sender and the receiver use designated (trusted) neighbors to carry their SOT, or the sender additively shares its message with its neighbors who perform the SOT on the sender's behalf. The structure of the TG guarantees that the neighbors can indeed perform SOTs with each other, i.e., while all direct SOTs may be aborted the "mediated" SOTs cannot be aborted without causing the overall protocol to abort.

In the proof we use the following multi-sender variant of global commitment which can be constructed from $\mathcal{F}^n_{\mathsf{COM}}$ by the Universal Composability Theorem of [11].

Functionality $\mathcal{F}^n_{\mathsf{MCOM}}$

$\mathcal{F}^n_{\mathsf{MCOM}}$ proceeds as follows, running with security parameter λ, parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are ignored.

- When receiving $(\texttt{commit}, m \in \{0,1\}^\lambda)$ from party P_i, store (P_i, m) and send $(\texttt{receipt commit}, \mathsf{P}_i)$ to all parties. Ignore further messages $(\texttt{commit}, \cdot)$ from P_i.
- When receiving $(\texttt{open})$ from party P_i and (P_i, m) is stored, send $(\texttt{open}, \mathsf{P}_i, m)$ to $\mathcal{S}$. Upon the next activation, send $(\texttt{output}, \mathsf{P}_i, m)$ to all parties and terminate. Ignore further input from P_i.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.

Theorem 4 (FCOT expansion). *Let $n = n(\lambda), t = t(\lambda), \beta := \lfloor n/(n-t) \rfloor + \lceil n/(n-t) \rceil - 2$ s.t. $n/2 \leq t \leq n-2$ and $\binom{n}{\beta} \in \text{poly}(\lambda)$. There is an efficient protocol π_{FCOT} that statistically securely UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta_{\mathsf{SFE},f}, \mathcal{F}^n_{\mathsf{COM}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against environments that (maliciously) corrupt up to t parties. Formally, for some specific functionality $\mathcal{F}^\beta$ we get*

$$\{\mathcal{F}^2_{\mathsf{SMT}}, \mathcal{F}^\beta, \mathcal{F}^n_{\mathsf{COM}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\rightsquigarrow}_t \mathcal{F}^n_{\mathsf{FCOT}} \ . \tag{3}$$

Proof. First, note that $\mathcal{F}^n_{\mathsf{TG}}$ is realizable using only $\mathcal{F}^n_{\mathsf{BC}}$ by Lemma 6, so we explicitly use $\mathcal{F}^n_{\mathsf{TG}}$ in the protocol construction. Moreover, note that $\{\mathcal{F}^n_{\mathsf{COM}}\} \overset{\text{stat}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{MCOM}}$ by the UC theorem. Thus we use $\mathcal{F}^n_{\mathsf{MCOM}}$ as well, and to reduce the notational complexity of the protocol we omit the session ids when they are clear from the context.

Denote the set of parties by $P = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, ..., \mathsf{W}_{n-2}\}$. We require all honest parties to locally compute the postprocessed Trust Graph $G^* \leftarrow \phi(G)$ when obtaining G from $\mathcal{F}^n_{\mathsf{TG}}$. Moreover, we require π_{FCOT} to be abort-respecting as described in Definition 8. As such, we condition our analysis on the case where the overall TG is connected.
Before we give the formal description of the protocol we outline it intuitively. The OT phase goes as follows:

1. The sender draws two mask bits at random and commits to them globally.
2. The sender creates two threshold sharings of its two masked messages.
3. The sender commits globally to each share individually.
4. The sender inputs its shares into the canonically smallest (not yet aborted) SOT-setup that contains both the sender and the receiver.
5. The receiver creates a threshold sharings of its choice bit.
6. The receiver commits globally to each share individually.
7. The receiver inputs its sharing into the canonically smallest (not yet aborted) SOT-setup that contains both the sender and the receiver.
8. The receiver obtains the chosen (masked) message.
9. All parties in the setup obtain some probing shares of (the sharings of) the masked messages and the choice bit.
10. The sender and the receiver open the global commitments corresponding to the shares at the probed indices.
11. The parties which participate in the SOT-setup verify that the opened shares are consistent with the ones obtained from the SOT-setup.
    11.1. If all shares are consistent, the sender opens the masks. The receiver obtains the chosen unmasked message. The other parties in the setup output a receipt, followed by all other parties (outside the setup).
    11.2. If any shares are inconsistent, then all parties in the setup declare a conflict with the resp. offender and its loyalists.[25] The current setup is aborted, its subgraph disconnected, and the (canonically) next setup is used with fresh sharings, i.e., the protocol goes back to step 1.
    11.3. If no setups containing sender and receiver are left, the protocol goes into fallback mode.

The fallback protocol uses the same basic idea with the following modifications:

- If the sender and the receiver have distance $\beta + 1$ in G and the receiver has exactly h neighbors (including itself), then the sender performs an SOT with the canonically smallest neighbor of the receiver. (In this case all neighbors of the receiver are honest.) Otherwise, if the receiver has more than h neighbors, the sender additively secret-shares its messages and gives them to its neighbors (who have distance β to R) who then perform SOTs with the receiver. The

[25] Recall this behavior is called abort-respecting in Definition 8, specifically Rule 2.

receiver obtains all shares and reconstructs the chosen message. Moreover, because at least one of the suited neighbors of the honest sender is honest the privacy of the non-chosen message is guaranteed.

On the other hand, if the sender and the receiver have distance $\beta + 2$, the sender creates a somewhat more intricate secret sharing and also gives the shares to suited neighbors who perform SOTs with suited neighbors of the receiver instead of the receiver directly.

- If the sender and the receiver have distance $\beta + 1$ in G and the receiver has exactly h neighbors, then the receiver gives its choice bit to its canonically smallest neighbor who then performs an SOT with the sender. Otherwise, if the receiver has more than h neighbors, then the receiver performs SOTs with the suited neighbors of the sender who each provide an additive share of the chosen message.

  On the other hand, if the sender and the receiver have distance $\beta + 2$, the receiver secret-shares it own choice bit and distributes the shares among suited neighbors who then perform SOTs with the sender's neighbors and forward the results to the receiver. Subsequently, the receiver can reconstruct the chosen message. Again, the privacy of the honest receiver's choice bit is guaranteed by the fact that at least one of the suited neighbors of the honest receiver is honest as well.

Furthermore, if the verification step in the fallback mode fails in all setups, then the overall Trust Graph becomes disconnected. The opening phase goes as follows:

1. The sender opens the global commitments to all shares of the resp. message.
2. The receiver open the global commitments to its shares.

Already the informal description of the protocol is non-trivial. To maintain the readability of the protocol and we describe the formal protocol only for the case where the sender and the receiver perform a direct SOT and argue the fallback case separately. We suggest the reader to keep Fig. 1 in mind when reading the protocol as it makes it easier to follow some arguments with a visual aid.

Now, we provide a more formal version of the proof. Let $h \coloneqq n - t$, and let $\ell \coloneqq n\lambda^2$ be the size of the applied secret sharing and let $\rho \coloneqq \lambda$ be the number of probing shares. Throughout the protocol let $\mathfrak{I}(S) \coloneqq \{M \subseteq P \mid |M| = \beta \wedge S \subseteq M \wedge (M, E^* \cap \binom{M}{\leq 2}) \text{ connected}\}$ be the set of subsets of parties containing parties S and whose subgraph is (still) connected. Note that this set might dynamically change during the protocol as setups are aborted. We use the number $\iota \coloneqq |\mathfrak{I}(\{\mathsf{S}, \mathsf{R}\})|$ as a counter. Everytime ι decreases the protocol restarts, i.e., the protocol informally goes to step 1. For each two parties $\mathsf{P}, \mathsf{P}' \in P$ let $M_{\mathsf{P},\mathsf{P}'} \coloneqq \min \mathfrak{I}(\{\mathsf{P}, \mathsf{P}'\})$ be the canonically smallest set of parties containing P, P' whose subgraph is (still) connected. Also, we denote the setup $\mathcal{F}_{\mathsf{SOT}}^{\beta}$ on parties $M_{\mathsf{P},\mathsf{P}'}$ with session id ι by $\mathcal{T}_{\mathsf{P},\mathsf{P}',\iota}$.

Protocol π_{FCOT} (simplified)

The protocol π_{FCOT} proceeds as follows, running with parties $P = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, ..., \mathsf{W}_{n-2}\}$, malicious parties $C \subseteq P$, adversary $\mathcal{A}$ and environment $\mathcal{Z}$. Messages not covered here are ignored.

- **In the first round** each party stores $G^*_{\text{ref}} \leftarrow (P, \emptyset)$, $\iota_{\text{ref}} \leftarrow |\mathfrak{I}(\{\mathsf{S}, \mathsf{R}\})|$ and $D_{\mathsf{S}}, D_{\mathsf{R}} \leftarrow \emptyset$.
- **Each round** each party P_i inputs $(\texttt{query})$ into $\mathcal{F}^n_{\mathsf{TG}}$ to obtain the TG in the next round.
- **On output** $G = (P, E)$ from $\mathcal{F}^n_{\mathsf{TG}}$, the party P computes the new $\iota \leftarrow |\mathfrak{I}(\{\mathsf{S}, \mathsf{R}\})|$. If $\iota < \iota_{\text{ref}}$, it deletes all previously received messages and updates the stored $G^*_{\text{ref}} \leftarrow \phi(G)$ and $\iota_{\text{ref}} \leftarrow \iota$.
- **On input** $(\texttt{choice}, c)$ from $\mathcal{Z}$, the receiver R creates a sharing $\gamma \leftarrow \text{Share}_{\ell,2\ell}(c)$ and commits to each share γ_κ by inputting $(\texttt{commit}, (\texttt{choice share}, \kappa, \gamma_\kappa))$ into $\mathcal{F}^n_{\mathsf{MCOM}}$. Then R inputs its shares $(\texttt{choice}, \mathsf{S} \to \mathsf{R}, c)$ into the SOT-setup $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$. The receiver ignores further inputs of this type.
- **On output** $(\texttt{receipt commit}, \mathsf{R})$ from $\mathcal{F}^n_{\mathsf{MCOM}}$ for all shares $\mu_\kappa : \kappa \in [2\ell]$, the party P outputs $(\texttt{receipt choice})$ if it did not output $(\texttt{receipt messages})$ before.
- **On input** $(\texttt{messages}, m^0, m^1)$ from $\mathcal{Z}$, the sender S samples masks $w^0, w^1 \leftarrow \{0,1\}$ and commits globally to $(\texttt{mask}, w^0, w^1)$. Then S creates a sharing $\mu^b \leftarrow \text{Share}_{\ell,2\ell}(m^b \oplus w^b)$ for both $b \in \{0,1\}$ and commits to each share μ^b_κ by inputting $(\texttt{commit}, (\texttt{message share}, b, \kappa, \mu^b_\kappa))$ into $\mathcal{F}^n_{\mathsf{MCOM}}$. Then S inputs its shares $(\texttt{messages}, \mathsf{S} \to \mathsf{R}, \mu^0, \mu^1)$ into the SOT-setup $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$. The sender ignores further inputs of this type from $\mathcal{Z}$.
- **On output** $(\texttt{receipt commit}, \mathsf{S})$ from $\mathcal{F}^n_{\mathsf{MCOM}}$ for all sessions containing the shares $\mu^b_\kappa : \kappa \in [2\ell], b \in \{0,1\}$ and the masks w^0, w^1, the party P outputs $(\texttt{receipt messages})$ if it did not output $(\texttt{receipt choice})$ before.
- **On output** $(\texttt{output}, \mathsf{S} \to \mathsf{R}, \mu^c)$ from $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$, the receiver R stores μ^c.
- **On output** $(\texttt{output}, \mathsf{S} \to \mathsf{R}, \nu, \widetilde{\mu}^0_\nu, \widetilde{\mu}^1_\nu, \widetilde{\gamma}_\nu)$ from $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$, the party P stores $(\nu, \widetilde{\mu}^0_\nu, \widetilde{\mu}^1_\nu, \widetilde{\gamma}_\nu)$. The sender and the receiver open all their global commitments corresponding to the probed shares ν by inputting $(\texttt{open})$ into $\mathcal{F}^n_{\mathsf{MCOM}}$ for each session containing $\kappa \in \nu$.
- **On output** $(\texttt{open}, \mathsf{S}, \mu^b_\kappa)$ from $\mathcal{F}^n_{\mathsf{MCOM}}$ for all $\kappa \in \nu$ and $b \in \{0,1\}$, then
  - if $\widetilde{\mu}^b_\nu \neq \mu^b_\nu$ for some $b \in \{0,1\}$, the setup $\mathcal{T}_{\mathsf{S},\mathsf{R}}$ is effectively aborted[a] with $(\texttt{abort}, \mathsf{S})$, i.e., the honest parties apply Rule 2 of Definition 8 to disconnect the subgraph on $M_{\mathsf{S},\mathsf{R}}$,
  - if $\widetilde{\gamma}_\nu \neq \gamma_\nu$, the setup $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$ is effectively aborted[b] with $(\texttt{abort}, \mathsf{R})$, i.e., the honest parties apply Rule 2 of Definition 8 to disconnect the subgraph on $M_{\mathsf{S},\mathsf{R}}$,
  - in any of the above cases, the sender and receiver (after the subgraph is disconnected) act as on input $(\texttt{messages}, m^0, m^1)$ and $(\texttt{choice}, c)$ respectively, i.e., the protocol restarts,
  - otherwise P_i broadcasts $(\texttt{receipt transfer})$.

Protocol π_{FCOT} (simplified) (cont'd)

- **On output** $(\texttt{output}, \mathsf{P}, (\texttt{receipt transfer}))$ from $\mathcal{F}^n_{\mathsf{BC}}$ for all $\mathsf{P} \in M_{\mathsf{S},\mathsf{R}}$, the sender opens the global commitment to the initial masks w^0, w^1.
- **On output** $(\texttt{open}, \mathsf{S}, (\texttt{mask}, w^0, w^1))$ from $\mathcal{F}^n_{\mathsf{MCOM}}$, any party $\mathsf{P}_i \neq \mathsf{R}$ outputs $(\texttt{receipt transfer})$ and stores (w^0, w^1). The receiver reconstructs $m^c \leftarrow w^c \oplus \mathrm{Recover}_{\ell,2\ell}(\mu^c)$ (using the previously obtained μ^c) and outputs $(\texttt{output}, m^c)$.
- **On input** $(\texttt{open message}, b)$ from $\mathcal{Z}$, the sender S opens the global commitments to the shares $(b, \kappa, \mu^b_\kappa)$ for each $\kappa \in [2\ell]$.
- **On output** $(\texttt{open}, \mathsf{S}, (\texttt{message share}, b, \kappa, \mu^b_\kappa))$ from $\mathcal{F}^n_{\mathsf{MCOM}}$ for all $\kappa \in [2\ell]$, then
  - P_i aborts with $(\texttt{abort}, \mathsf{S})$, if $\mathrm{Recover}_{\ell,2\ell}(\mu^b) = \bot$, or otherwise
  - P_i outputs $(\texttt{open message}, b, \widetilde{m}^b)$ where $\widetilde{m}^b \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu^b) \oplus w^b$.
- **On input** $(\texttt{open choice})$ from $\mathcal{Z}$, the receiver R opens the global commitments to the shares (κ, γ_κ) for each $\kappa \in [2\ell]$.
- **On output** $(\texttt{open}, \mathsf{R}, (\texttt{choice share}, \kappa, \gamma_\kappa))$ from $\mathcal{F}^n_{\mathsf{MCOM}}$for all $\kappa \in [2\ell]$, then
  - P_i aborts with $(\texttt{abort}, \mathsf{R})$, if $\mathrm{Recover}_{\ell,2\ell}(\gamma) = \bot$, or otherwise
  - P_i outputs $(\texttt{open choice}, \widetilde{c})$ where $\widetilde{c} \leftarrow \mathrm{Recover}_{\ell,2\ell}(\gamma)$.

[a] All participants know that the sender broadcasted inconsistent shares.
[b] All participants know that the receiver broadcasted inconsistent shares.

Before analyzing the protocol in detail we give a simulator for the canonical dummy adversary. The simulator runs a simulated protocol where it executes the protocol code for all non-corrupted parties. Whenever the environment activates an honest dummy party, the simulator is notified and simulates the party's protocol code. Any message from and to malicious parties are forwarded to the simulated setup functionalities.
Note that at most $n\rho$ many shares of the sender's and receiver's committed sharings are probed.

Simulator for π_{FCOT} (simplified)

- **On output** $(\texttt{receipt messages})$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ or **on output** $(\texttt{receipt transfer})$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ after $(\texttt{receipt choice})$, simulator $\mathcal{S}$ gives input $(\texttt{messages}, 0, 0)$ to the simulated sender S'.
- **On output** $(\texttt{receipt messages})$ from all simulated (honest) parties, the simulator $\mathcal{S}$ inputs $(\texttt{messages}, \widetilde{m}^0 \oplus w^0, \widetilde{m}^1 \oplus w^1)$ to the ideal $\mathcal{F}^n_{\mathsf{FCOT}}$ in the name of S where $\widetilde{m}^b \leftarrow \mathrm{Recover}_{\ell,2\ell}(\widetilde{\mu}^b) \oplus w^b$. Here, (w^0, w^1) are the masks and $\widetilde{\mu}^b$ is the sharing that the malicious sender previously input into the simulated commitment setup $\mathcal{F}^n_{\mathsf{MCOM}}$.
- **On output** $(\texttt{receipt choice})$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ or **on output** $(\texttt{receipt transfer})$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ after $(\texttt{receipt messages})$, simulator $\mathcal{S}$ gives input $(\texttt{choice}, 0)$ to the simulated receiver R'.

Simulator for π_{FCOT} (simplified) (cont'd)

- **On output** (`receipt choice`) from all simulated (honest) parties, the simulator $\mathcal{S}$ (`choice`, $\widetilde{c}$) to the ideal $\mathcal{F}^n_{\mathsf{FCOT}}$ in the name of R where $\widetilde{c} \leftarrow \text{Recover}_{\ell,2\ell}(\widetilde{\gamma})$ and $\widetilde{\gamma}$ is the sharing that the malicious receiver previously input into the simulated commitment setup $\mathcal{F}^n_{\mathsf{MCOM}}$.
- **On output** (`open choice`, $\widetilde{c}$) from $\mathcal{F}^n_{\mathsf{FCOT}}$, the simulator $\mathcal{S}$ gives local input (`open`) to the simulated receiver R' which will open the remaining setups $\mathcal{F}^n_{\mathsf{MCOM}}$. However, the simulator equivocates the remaining simulated $\mathcal{F}^n_{\mathsf{MCOM}}$ containing share $\widetilde{\gamma}_\kappa$ for the indices $[2\ell]\setminus\nu$. That is, $\gamma' \leftarrow \text{Share}_\ell(\widetilde{c})$ while $\gamma'_\nu = \widetilde{\gamma}_\nu$ where ν is the set of already probed indices. To this end the simulated $\mathcal{F}^n_{\mathsf{MCOM}}$ on input (`open`) from R sends (`open`, $(\kappa, \widetilde{\gamma}_\kappa)$) to all parties and the simulated dummy adversary. This is possible because at most $n\rho$ shares have been probed but the simulator only needs to equivocate $\ell + 1 = n\lambda^2 + 1 \leq n(2\lambda^2 - \lambda) = 2\ell - n\rho$ shares.
- **On output** (`open choice`, $\widetilde{c}$) from all simulated parties, the simulator sends (`open choice`) to the ideal $\mathcal{F}^n_{\mathsf{FCOT}}$ in the name of R.
- **On output** (`open message`, $b, \widetilde{m}^b$) from $\mathcal{F}^n_{\mathsf{FCOT}}$, the simulator $\mathcal{S}$ gives local input (`open`) to the simulated sender S' which will open the remaining setups $\mathcal{F}^n_{\mathsf{MCOM}}$. Again, the simulator equivocates the remaining simulated $\mathcal{F}^n_{\mathsf{MCOM}}$ containing share $\widetilde{\mu}^b_\kappa$ for each $\kappa \in [\ell] \setminus \nu$. The same argument as for the equivocation of the receiver's shares applies.
- **On output** (`open message`, $b, \widetilde{m}^b$) from all simulated parties, the simulator inputs (`open message`, b) to the ideal $\mathcal{F}^n_{\mathsf{FCOT}}$ in the name of S.
- **On input** (`abort`, C') for any simulated setup with parties $M \subseteq P$ and $\emptyset \neq C' \subset M \cap C$, the simulator aborts that instance by forwarding (`abort`, C') to it.
- **On output** (`abort`, C') from all simulated parties, simulator $\mathcal{S}$ inputs (`abort`, C') into $\mathcal{F}^n_{\mathsf{FCOT}}$.

From the description of the simulator it is apparent that the honest dummy parties output the receipt resp. the opened value exactly when the simulated (honest) parties would output the receipt resp. opened value. It is also clear from the simulator's description that for an honest sender resp. receiver the simulator can equivocate its simulated setup functionalities to let the simulated parties output $\widetilde{m}^0, \widetilde{m}^1$ resp. $\widetilde{\gamma}$ which the simulator receives from $\mathcal{F}^n_{\mathsf{FCOT}}$ at the start of the opening phase. In particular, because the simulator extracts its inputs $\widetilde{m}^0, \widetilde{m}^1$ and $\widetilde{\gamma}$ from whatever inputs are encoded in the sharings in the global commitments and all parties output exactly the opened value from the global commitments, the opened messages resp. choice bit are equal in the real and ideal run.
Henceforth, we focus on proving that

- When all parties output (`receipt transfer`), then the inputs of the last used setup $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$ are indeed equal to the inputs encoded in the sharings contained in the global commitments.

- All (honest) parties eventually output (`receipt transfer`) or the protocol aborts.

To this end we make some observations.

Claim 6. *When all (honest) parties output* (`receipt transfer`) *in the real/simulated protocol and the last used setup $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$ contains some honest party, the inputs encoded in the sharings of $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$, i.e., $\widetilde{\mu}^b$ and $\widetilde{\gamma}$, and the globally committed sharings $\widehat{\mu}^b$ and $\widehat{\gamma}$ are equal with overwhelming probability.*

We prove Claim 6 similarly to Claim 2. Suppose the choice bit encoded in the sharing of $\mathcal{T}_{\mathsf{S},\mathsf{R},\iota}$, i.e., $\widetilde{\gamma}$, and the globally committed sharing $\widehat{\gamma}$ differ. Because $\mathrm{Recover}_{\ell,2\ell}(\widetilde{\gamma}) \neq \perp$ and $\mathrm{Recover}_{\ell,2\ell}(\widehat{\gamma}) \neq \perp$ there exists a set $\widetilde{\nu} \in \binom{[2\ell]}{\ell+1}$ of differing indices s.t. $\forall \kappa \in \widetilde{\nu} : \widetilde{\gamma}_\kappa \neq \widehat{\gamma}_\kappa$. Furthermore $\widetilde{\nu}$ must be disjoint from the uniformly random probing indices $\nu \leftarrow \binom{[2\ell]}{\rho}$, otherwise the parties $M_{\mathsf{S},\mathsf{R},\iota} \cap H \neq \emptyset$ would have declared a conflict with the sender and/or the receiver. Lemma 5 bounds the probability that no differing share is probed by

$$\Pr_{\nu \leftarrow \binom{[2\ell]}{\rho}}[\widetilde{\nu} \cap \nu = \emptyset] \overset{\text{Eq. (48)}}{\leq} 2^{-\rho(\ell+1)/2\ell} \leq 2^{-\lambda/2} \in \mathrm{negl}(\lambda) \ . \tag{50}$$

The same argument applies to the two sharings for the sender's messages. Consequently, by the union bound over all possible commitments (for the maximum of n^2 restarts) we know that each of the encoded messages are equal in the SOT-setup and in the global commitments with probability at least $1 - 3n^4 2^{-\lambda/2}$.

For exhibition, we can now give a series of hybrid games:

$\mathcal{H}_1$ This game is the real execution where all parties follow the protocol.

$\mathcal{H}_2$ This game is the same as before except that there is an ideal dummy functionality $\mathcal{F}^n_{\mathsf{FCOT}}$ which is connected to (honest) dummy parties which obtain the same local inputs as the honest parties in the real protocol. The real adversary $\mathcal{A}$ is replaced with a simulator $\mathcal{S}$ that behaves exactly as $\mathcal{A}$ and, if the sender is malicious, the simulator $\mathcal{S}$ extracts the sender's messages $(\texttt{messages}, m^0, m^1)$ for $\mathcal{F}^n_{\mathsf{FCOT}}$ from the real protocol execution, i.e., from the sharings μ^0, μ^1 input into the commitment setup. If the sender is honest, then the honest dummy sender forwards its input $(\texttt{messages}, m^0, m^1)$ to $\mathcal{F}^n_{\mathsf{FCOT}}$. If the receiver is malicious, the simulator $\mathcal{S}$ extracts the receiver's choice bit $(\texttt{choice}, c)$ for $\mathcal{F}^n_{\mathsf{FCOT}}$ from the real protocol execution, i.e., from the sharing γ input into the commitment setup. If the sender is honest, then the honest dummy sender forwards its input $(\texttt{choice}, c)$ to $\mathcal{F}^n_{\mathsf{FCOT}}$.
The output of the dummy parties is ignored, i.e., the game's output is still exactly the same as before.

$\mathcal{H}_3$ This game is the same as before except that this game aborts, iff the (opened) message output by the honest parties differs from the output of the dummy parties interacting with the ideal dummy functionality.
This event of differing outputs occurs only negligibly often, as argued in Claim 6. Hence, the game's output behavior is statistically close to the previous one.

$\mathcal{H}_4$ This game is the same as before except that now the environment interacts with the the dummy parties of the ideal functionality instead of the parties in the (real) execution output. (Still the parties of the protocol execution get the same input as the dummy parties, i.e., they behave as before. The only difference is that their output is ignored.)

Due to the abort condition established in the previous game these outputs are identical, hence the output of this game is exactly the same as on the previous game.

$\mathcal{H}_5$ This game is the same as before except that the parties in the protocol now longer obtain the environment's input but are instead simulated by the simulator $\mathcal{S}$. In particular, if the simulator gets (`receipt messages`), or (`receipt transfer`) after (`receipt choice`) from the ideal $\mathcal{F}^n_{\mathsf{COM}}$ it simulates the honest sender with input ($\texttt{messages}, 0, 0$). If the simulator gets (`receipt choice`), or (`receipt transfer`) after (`receipt messages`) from the ideal $\mathcal{F}^n_{\mathsf{FCOT}}$ it simulates the honest receiver with input ($\texttt{choice}, 0$). This game is the ideal execution.

This game and the previous game have exactly the same output behavior because the (honest) dummy parties's output depends solely on the ideal functionality $\mathcal{F}^n_{\mathsf{FCOT}}$ and not on the local input of the simulated sender.

It remains to show that all parties eventually output (`receipt transfer`) if the protocol does not abort. In the above description of the protocol we have assumed that at least one setup containing both the sender and the receiver will not be aborted. If this was the case we would already have concluded the proof.

Fallback protocol. In the remaining part we argue that even if the sender and the receiver cannot perform a *direct* SOT—in the fallback case—the protocol still works. The simulator for the fallback case is essentially the same as for the simplified version except that the protocol code of the sender and receiver also covers the above cases when their input has to be delegated to suited neighbors. We stress that in the fallback mode, either the sender, or the receiver, or both are corrupted. We consider the following cases one by one.

1. The distance between sender and receiver in G is $\beta + 1$, and R has exactly h neighbors; then all neighbors of an honest receiver are honest as well. The receiver simply gives its choice bit to the smallest neighbor who then performs the SOT with the sender.
2. The distance between sender and receiver in G is $\beta + 1$, and the receiver has strictly more than h neighbors. Then the sender secret-shares its messages and gives one share to each neighbor (with distance β to R) who then performs an SOT with the receiver.
3. The distance between sender and receiver in G is $\beta + 2$. Then the sender secret-shares its messages and gives one share to each neighbor (with distance $\beta + 1$ to R) who then performs an SOT with the receiver's neighbors. Similarly, the receiver secret-shares its choice bit and gives one share to each neighbor (with distance $\beta + 1$ to S) who then performs an SOT with the sender's neighbors.

Furthermore, if all setups containing some neighbor $\mathsf{N_S} \in N_\mathsf{S}$ and some neighbor $\mathsf{N_R} \in N_\mathsf{R}$, then Corollary 3 states that the overall Trust Graph must be disconnected, i.e., the protocol successfully aborts.

Claim 7. *The distance between the sender and receiver in G is at most $\beta + 2$, i.e., $\Delta_G(\mathsf{S}, \mathsf{R}) \leq \beta + 2$.*

Proof. For contradiction suppose $\Delta_G(\mathsf{S}, \mathsf{R}) \geq \beta + 3$. By recalling Corollary 3 and Fig. 1 we count at least

$$n' := \lceil(\beta + 3)/2\rceil + \lfloor(\beta + 3)/2\rfloor(h - 1) \tag{51}$$

parties, i.e., the L- and S-segments of any shortest path between S and R. Now we show that $n' > n$.

- If $n/h \in \mathbb{N}$, then $n' = (n/h + 1) + (n/h)(h - 1) = n + 1$.
- If $n/h \notin \mathbb{N}$, then $n' = (\lfloor n/h\rfloor + 1) + (\lfloor n/h\rfloor + 1)(h - 1) = \lceil n/h\rceil h > n$.

■

Claim 8. *If $\Delta_G(\mathsf{S}, \mathsf{R}) = \beta + 1$ and $|\,\mathrm{N}_G(\mathsf{R})| = h$, then the receiver gives its choice bit to its canonically smallest neighbor who then performs the SOT with the sender.*

Proof. First, note that if an honest party has h neighbors (including itself), then all neighbors are honest as well. Hence the privacy of the honest sender's (resp. receiver's) input is guaranteed. Moreover, all neighbors of the sender (resp. receiver) must have distance β to the receiver (resp. sender), i.e., any neighbor can perform an SOT with the receiver (resp. sender). ■

Claim 9. *If $\Delta_G(\mathsf{S}, \mathsf{R}) \geq \beta + 1$ and $|\,\mathrm{N}_G(\mathsf{R})| > h$, then $|\,\mathrm{N}_G(\mathsf{S})| \leq 2h - 2$.*

Proof. For contradiction suppose $|\,\mathrm{N}_G(\mathsf{S})| \geq 2h - 1$, then we count at least

$$\begin{aligned} n' &:= \underbrace{(2h-1)}_{\mathrm{N}_G(\mathsf{S})} + \underbrace{\lceil(\beta - 3)/2\rceil}_{L\text{-segments}} + \underbrace{\lfloor(\beta - 3)/2\rfloor(h - 1)}_{S\text{-segments}} + \underbrace{h+1}_{\mathrm{N}_G(\mathsf{R})} \\ &= 3h + \lceil(\beta - 3)/2\rceil + \lfloor(\beta - 3)/2\rfloor(h - 1) \end{aligned} \tag{52}$$

parties. Now we show that $n' > n$.

- If $n/h \in \mathbb{N}$, then $n' = 3h + (n/h - 2) + (n/h - 3)(h - 1) = n + 1$.
- If $n/h \notin \mathbb{N}$, then $n' = 3h + (\lfloor n/h\rfloor - 2) + (\lfloor n/h\rfloor - 2)(h - 1) = \lceil n/h\rceil h > n$.

As a consequence, each subset of at least $h - 1$ neighbors of the honest sender (resp. receiver, excluding themselves) contains at least one honest party because the honest sender (resp. receiver) has at least $h - 1$ honest neighbors (excluding itself). ■

Claim 10. *If $\Delta_G(\mathsf{S}, \mathsf{R}) = \beta + 1$ and $|\,\mathrm{N}_G(\mathsf{S})| \leq 2h - 2$, then the sender can additively secret-share its messages and distribute the shares among its neighbors with distance β to the receiver. These neighbors can then perform the SOT with the receiver. In other words, at least one neighbor (of the honest sender) with distance β to the receiver is honest.*

Proof. This procedure protects the honest sender's privacy because—as Claim 9 just established—at least one additive share is held by an honest neighbor. Because in thefallback case, if the sender is honest, the receiver must be malicious, we do not not need to worry about the receiver obtaining the correct chosen message. ■

Claim 11. *If $\Delta_G(\mathsf{S},\mathsf{R}) = \beta + 2$, then $|\,\mathrm{N}_G(\mathsf{S})|, |\,\mathrm{N}_G(\mathsf{R})| \leq 2h - 2$.*

Proof. For contradiction suppose $|\,\mathrm{N}_G(\mathsf{S})| \geq 2h - 1$, then we count at least

$$n' := (2h-1) + \lceil \beta/2 \rceil + \lfloor \beta/2 \rfloor (h-1) \tag{53}$$

parties. Now we show that $n' > n$.

- If $n/h \in \mathbb{N}$, then $n' = 2h-1+(n/h-1)+(n/h-1)(h-1) = n+h-1 \geq n+1$.
- If $n/h \notin \mathbb{N}$, then $n' = 2h-1+\lfloor n/h \rfloor + (\lfloor n/h \rfloor - 1)(h-1) = \lceil n/h \rceil h > n$.

The analogous argument applies for $|\,\mathrm{N}_G(\mathsf{R})| \geq 2h-1$. ■

Claim 12. *If $\Delta_G(\mathsf{S},\mathsf{R}) = \beta + 2$ and $|\,\mathrm{N}_G(\mathsf{S})|, |\,\mathrm{N}_G(\mathsf{R})| \leq 2h - 2$, the sender can additively secret-share its messages and distribute the shares among its neighbors with distance $\beta + 1$ to R. Moreover, the receiver[26] can additively secret-share its choice bit and distribute the shares among its neighbors with distance $\beta + 1$ to S. These neighbors can then perform the SOTs with each other and forward the results.*

Proof. More precisely, let

$$\begin{aligned} N_{\mathsf{S}}^{\beta+1} &:= \{\mathsf{N} \in \mathrm{N}_G(\mathsf{S}) \mid \Delta_G(\mathsf{N},\mathsf{R}) = \beta+1\} = \left\{\mathsf{N}_1^{\mathsf{S}}, ..., \mathsf{N}_{a'_{\mathsf{S}}}^{\mathsf{S}}\right\}, \qquad a'_{\mathsf{S}} := \left|N_{\mathsf{S}}^{\beta+1}\right| \\ N_{\mathsf{R}}^{\beta+1} &:= \{\mathsf{N} \in \mathrm{N}_G(\mathsf{R}) \mid \Delta_G(\mathsf{N},\mathsf{S}) = \beta+1\} = \left\{\mathsf{N}_1^{\mathsf{R}}, ..., \mathsf{N}_{a'_{\mathsf{R}}}^{\mathsf{R}}\right\}, \qquad a'_{\mathsf{R}} := \left|N_{\mathsf{R}}^{\beta+1}\right| \end{aligned} \tag{54}$$

be the *suited* neighbors of the sender (resp. receiver). Note that $a'_{\mathsf{S}}, a'_{\mathsf{R}} \geq h-1$. The sender creates an additive sharing $\mu^b \in \{0,1\}^{a'_{\mathsf{S}}}$ per message s.t. $m^b = \bigoplus_{i_{\mathsf{S}} \in [a'_{\mathsf{S}}]} \mu^b_{i_{\mathsf{S}}}$. For each $i_{\mathsf{S}} \in [a'_{\mathsf{S}}]$ the sender sets randomly

$$\begin{aligned} \forall i_{\mathsf{R}} \in [a'_{\mathsf{R}} - 1] : z^0_{i_{\mathsf{S}}, i_{\mathsf{R}}} &\leftarrow \{0,1\} \\ z^0_{i_{\mathsf{S}}, a'_{\mathsf{R}}} &:= \mu^0_{i_{\mathsf{S}}} \oplus \bigoplus_{i_{\mathsf{R}} < a'_{\mathsf{R}}} z^0_{i_{\mathsf{S}}, i_{\mathsf{R}}} \\ \forall i_{\mathsf{R}} \in [a'_{\mathsf{R}}] : z^1_{i_{\mathsf{S}}, i_{\mathsf{R}}} &:= \mu^0_{i_{\mathsf{S}}} \oplus \mu^1_{i_{\mathsf{S}}} \oplus z^0_{i_{\mathsf{S}}, i_{\mathsf{R}}} \end{aligned} \tag{55}$$

and gives the sharings $z^0_{i_{\mathsf{S}}}, z^1_{i_{\mathsf{S}}}$ to $\mathsf{N}^{\mathsf{S}}_{i_{\mathsf{S}}}$. In turn, the receiver creates an additive sharing $\gamma_{i_{\mathsf{R}}} \in \{0,1\}^{a'_{\mathsf{R}}}$ for each neighbor $\mathsf{N}^{\mathsf{R}}_{i_{\mathsf{R}}}$ s.t. $c = \bigoplus_{i_{\mathsf{R}} \in [a'_{\mathsf{R}}]} \gamma_{i_{\mathsf{R}}, i_{\mathsf{S}}}$. The neighbor $\mathsf{N}^{\mathsf{S}}_{i_{\mathsf{S}}}$ (in the role of the sender) performs an SOT with each neighbor $\mathsf{N}^{\mathsf{R}}_{i_{\mathsf{R}}}$ (in the role of the receiver) where the message sharings are $z^0_{i_{\mathsf{S}}, i_{\mathsf{R}}}, z^1_{i_{\mathsf{S}}, i_{\mathsf{R}}}$ and the choice bit sharing is $\gamma_{i_{\mathsf{R}}, i_{\mathsf{S}}}$. Consequently, the neighbor $\mathsf{N}^{\mathsf{R}}_{i_{\mathsf{R}}}$ obtains the share $z^{\gamma_{i_{\mathsf{R}}, i_{\mathsf{S}}}}_{i_{\mathsf{S}}, i_{\mathsf{R}}}$ from

[26] This setup bares some similarity to the minimal protocol of Wolf and Wullschleger [43] which shows the symmetry of OT.

the SOT with $\mathsf{N}^{\mathsf{S}}_{i_\mathsf{S}}$ and forwards it to R. Finally, R can reconstruct

$$\begin{aligned}
\bigoplus_{i_\mathsf{S}\in[a'_\mathsf{S}],i_\mathsf{R}\in[a'_\mathsf{R}]} z^{\gamma_{i_\mathsf{R},i_\mathsf{S}}}_{i_\mathsf{S},i_\mathsf{R}} &= \bigoplus_{i_\mathsf{S}\in[a'_\mathsf{S}]}\ \bigoplus_{i_\mathsf{R}\in[a'_\mathsf{R}]} \left(\gamma_{i_\mathsf{R},i_\mathsf{S}}\left(\mu^0_{i_\mathsf{S}}\oplus\mu^1_{i_\mathsf{S}}\right)\oplus z^0_{i_\mathsf{S},i_\mathsf{R}}\right)\\
&= \bigoplus_{i_\mathsf{S}\in[a'_\mathsf{S}]}\left(\left(\mu^0_{i_\mathsf{S}}\oplus\mu^1_{i_\mathsf{S}}\right)\left(\bigoplus_{i_\mathsf{R}\in[a'_\mathsf{R}]}\gamma_{i_\mathsf{R},i_\mathsf{S}}\right)\oplus\bigoplus_{i_\mathsf{R}\in[a'_\mathsf{R}]} z^0_{i_\mathsf{S},i_\mathsf{R}}\right)\\
&= \bigoplus_{i_\mathsf{S}\in[a'_\mathsf{S}]}\left(\left(\mu^0_{i_\mathsf{S}}\oplus\mu^1_{i_\mathsf{S}}\right)c\oplus\mu^0_{i_\mathsf{S}}\right)\\
&= \bigoplus_{i_\mathsf{S}\in[a'_\mathsf{S}]}\mu^c_{i_\mathsf{S}}\\
&= m^c
\end{aligned} \tag{56}$$

very efficiently by adding all obtained shares. The privacy of the honest sender's message (resp. the honest receiver's choice) bit is guaranteed by Claim 11 because at least one out of the at least $h-1$ suited neighbors is honest.
Naturally, the sender and the receiver commit globally to each of their shares (here $z^0_{i_\mathsf{S},i_\mathsf{R}}$, $z^1_{i_\mathsf{S},i_\mathsf{R}}$ and $\gamma_{i_\mathsf{R},i_\mathsf{S}}$ instead of μ^0, μ^1 and γ) to enable the same verification as in the main protocol. ■

In the fallback protocol at most n^2 SOTs have to be called per restart, i.e., aborted setup, and at most n^2 setups can be aborted because each abort removes at least one edge from the TG. Overall, we find that at most $\mathcal{O}(n^4)$ setups will be called. □

D.3 Equivalence of FCOT, SFE and Correlated-Randomness

Lemma 8. *Let $n = n(\lambda)$. The functionalities $\mathcal{F}^n_{\mathsf{SFE},f}$, $\mathcal{F}^n_{\mathsf{FCOT}}$ and $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ are equally powerful. Formally,*

$$\mathcal{F}^n_{\mathsf{FCOT}} \overset{\text{stat}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{SFE},f} \overset{\text{perf}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}} \overset{\text{stat}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{FCOT}}\ . \tag{57}$$

The first construction is essentially the IPS-compiler with FCOT setups. The second construction follows directly from the fact that $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ is a special case of $\mathcal{F}^n_{\mathsf{SFE},f}$. The third construction follows from the MPC-completeness of the Correlated-Randomness model in [30].

The IPS-compiler. Before showing the construction $\mathcal{F}^n_{\mathsf{FCOT}} \overset{\text{stat}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{SFE},f}$ we first recall the so-called IPS-compiler [31] (see also [35] for reference).

The IPS-compiler provides n-party MPC from two-party OT against arbitrarily many malicious parties in the Anonymous Abort setting. Thereby, two separate protocols with different security guarantees are combined: The so-called outer protocol Π provides security against a malicious adversary but only for an honest majority (that is, $t < n/2$), while the inner protocol ρ is secure against arbitrarily many semi-honest parties. The resulting protocol Φ then inherits the best of both worlds; is remains secure against any number of malicious parties.

Instead of performing a single n-party MPC directly on their respective inputs, the n parties *simulate* the behavior of a larger number $m \in \mathcal{O}(n^2)$ of virtual parties. The high level idea is for the n parties to engage in a combined protocol Φ that lets them use the outer protocol Π to additively share their input with the m virtual parties, who then execute the inner protocol ρ based on these inputs. The inner protocol is *only* secure against (arbitrarily many) semi-honest parties but their right behavior is ensured by letting the combined protocol use the outer protocol to deploy a *watchlist* for each of the m inner parties. That way each of the n actual parties can pre-compute the to-be-received messages of the m simulated parties from the inner protocol. Thus misbehavior of any outside party is detected with high probability.

The m simulated parties of the inner protocol are generally called *servers*, the n actual parties are referred to as *clients*. This is due to the outer protocol Π making black-box use of the inner protocol ρ reminiscent of a client-server model. For each server, each client draws a long one-time pad. When a party sends its input to the i-th server, it also encrypts the message with successive parts of the corresponding one-time pad and broadcasts it on its *watchlist broadcast channel* of the i-th server. At the beginning of the combined protocol each client offers each other client a certain fraction of their own one-time pads via OT. Thus, a client P that is in possession of the one-time pad that another client P' uses for the i-th server can read all messages that P' inputs into the i-th server. If a party is in possession of all one-time pads used for the i-th server, it can read all messages input into the i-th server. This way the party knows the complete state of that server and hence can pre-compute the messages that server will receive in advance. If the messages the i-th server actually received deviate from the own pre-computation based on the watchlist broadcast channel then the affected server must be corrupted and the party can abort. This way the watchlist mechanism ensures that the servers execute the inner protocol correctly, hence it suffices for ρ to be secure only against semi-honest adversaries; either sufficiently many servers are correct or the computation is aborted.

The parameters for the secret sharing scheme are chosen such that on the one hand side the probability for unnoticed deviation from the inner protocol is negligible while on the other hand no information from the watched servers allows reconstruction of the clients actual inputs; for further details we refer to [31].

As an artifact of their security notion a party can *notice* malicious behavior but neither *identify* the cheater, nor *convince* other parties who do not have that server on their watchlist that misbehavior has occurred.

While Cohen and Lindell [17] already obtained MPC with IA by making non-black-box use of the GMW-compiler [25], we claim that the IPS compiler can also be modified so as to yield IA: By replacing two-party OT $\mathcal{F}^2_{\mathsf{OT}}$ with Fully Committed Oblivious Transfer $\mathcal{F}^n_{\mathsf{FCOT}}$, any client that detects the misbehavior can post hoc request the opening of all communication regarding the affected server and the corresponding one-time pads such that all parties can retrace the inner protocol and identify which party made inputs to the affected server that do not match the value on the watchlist broadcast channel:

Lemma 9 (FCOT $\rightsquigarrow_n$ SFE). *Let $n = n(\lambda)$. Let $f : (\{0,1\}^\lambda)^n \to (\{0,1\}^\lambda)^n$ be an efficiently uniformly computable family of functions. There is an efficient protocol π_{SFE} that statistically securely UC-realizes $\mathcal{F}^n_{\mathsf{SFE},f}$ in the $\{\mathcal{F}^n_{\mathsf{FCOT}}\}$-hybrid model against environments that (maliciously) corrupt up to t parties. Formally,*

$$\mathcal{F}^n_{\mathsf{FCOT}} \overset{\text{stat}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{SFE},f} \ . \tag{58}$$

Proof. Here, we prove that there exists a $\{\mathcal{F}^n_{\mathsf{FCOT}}\}$-hybrid protocol Φ that securely UC-realizes $\mathcal{F}^n_{\mathsf{SFE},f}$. Again, for arbitrary n, denote the set of parties by P.

We use the IPS-compiler [31] described above, which compiles two protocols Π and ρ into an $\{\mathcal{F}^2_{\mathsf{OT}}\}$-hybrid protocol Φ. Their result cannot be directly transferred into the setting of IA. However, we slightly modify their protocol in the following ways: (1) All communication in the new inner protocol is processed via FCOTs. (2) We replace $\mathcal{F}^2_{\mathsf{OT}}$-calls with to $\mathcal{F}^n_{\mathsf{FCOT}}$. (3) When a client P would abort in the original protocol [31] due to noticing misbehavior in server i, it instead publicly demands the opening of all communication corresponding to server i. After the opening, it holds that either there is a set of malicious parties C' that was identified by this procedure or that refused to open their communication,[27] then all honest parties send $(\texttt{conflict}, C')$ to $\mathcal{F}^n_{\mathsf{TG}}$. Or the opened messages indicate no malicious behavior on that server, in which case all honest parties send $(\texttt{conflict}, \mathsf{P})$ to $\mathcal{F}^n_{\mathsf{TG}}$. Note that Item (1) is without loss of generality as OT (hence also FCOT) suffices to set up authenticated channels, but allows opening the relevant communication upon accusation of misbehavior. Item (2) also does not change the behavior of the simulator as the extraction that Ishai, Prabhakaran, and Sahai [31] use for OT is compatible with the definition of FCOT. To prove security we can hence use essentially the same simulator that was also used by Ishai, Prabhakaran, and Sahai [31], which only has to be adjusted to incorporate the new Identifiable Abort criteria from Item (3).

Item (3) only changes behavior with respect to aborts, so if no aborts occur then simulation is exactly the same and the modified simulator learns as much information as the original simulator from [31]. The only adaptations to the simulator are with respect to detected misbehavior of parties: instead of merely forwarding the abort of a single party to the functionality our simulator must provide a set of corrupted parties C' to abort the ideal functionality $\mathcal{F}^n_{\mathsf{SFE},f}$, and all simulated parties in the protocol must provide output $(\texttt{abort}, C')$. The new protocol ensures this in the following way. The key to the new behavior is a parties message $(\texttt{challenge}, i)$ which a party can use to receive an explanation of the behavior regarding the i-th server. This can either be caused by the simulator directly in the name of an honest party after detecting misbehavior in

[27] The synchronous model allows parties to notice when a party denies opening.

the i-th server, or by a malicious party who is controlled by the environment. Yet both scenarios are handled equivalently by the simulator. The challenge causes all parties to open the FCOTs used to distribute their watchlist one-time pads and explain the messages that are related to the i-th instance of the inner protocol (that is, the i-th server). More precisely, we assume that for distributing the watchlist keys a $\binom{n^2\lambda}{\lambda}$-FCOT was used for each party which can be canonically constructed from $\binom{2}{1}$-FCOTs. Then each choice index in the $\binom{n^2\lambda}{\lambda}$-FCOT corresponds to multiple choice bits in the $\binom{2}{1}$-FCOTs in a priori known manner, hence all $\binom{2}{1}$-messages m_c neighbored with the i-th watchlist can be opened.

Consequently, all parties learn the complete in- and outcoming messages of the i-th server but no additional communication of any other server. Thus each party can retrace the complete computation of the i-th server and register any deviation from the protocol. Again, either the challenging party indeed identified a malicious message on the i-th server, then all parties notice this misbehavior and identify the malicious party P and abort with $(\mathtt{abort}, \mathsf{P})$. Or the challenging party lied about receiving a malicious message on the i-th server, which only happens if the calling party is really malicious; in which case all other parties will abort with $(\mathtt{abort}, \mathsf{P}')$ where P' corresponds to the party that sent challenged server i. Thus the simulator can extract the identity of a malicious party, and use it to abort the ideal functionality $\mathcal{F}^n_{\mathsf{SFE},f}$ in the name of that party. Note that even in the original simulator [31], after a misbehavior has been detected by the simulator it corrupts that server in the inner protocol; hence our induced changes leak no additional information to the adversary.

However, we must still ensure that opening all inputs of a single server does not violate the privacy of the clients in the outer protocol Π. In the following, we formalize this idea: Let n be the number of clients in the outer protocol. In the original paper [31] there are $m \in \Theta(n^2\lambda)$ servers. Each party gets to select λ watchlists from each party, such that each party can see all in- and outcoming communication of λ servers. In total, at most a fraction of $n\lambda/n^2\lambda = 1/n$ of all servers state is known by any set of parties. Because the used secret sharing requires a constant fraction of shares to reconstruct the original input, no coalition of parties can learn the input of another party. Now, if misbehavior occurs and the state of an additional server is opened any coalition of parties knows at most $\frac{n\lambda+1}{n^2\lambda} \leq \frac{2}{n}$, which is still less than a constant fraction.

Thus, with the induced changes not violating privacy and with the simulator being able to correctly simulate the modified protocol in the setting of Identifiable Abort we have proven our claim. $\square$

Instantiating the Correlated-Randomness model.

Lemma 10 (SFE $\rightsquigarrow_t$ Correlated-Randomness). *Let $n = n(\lambda)$. Let $\mathcal{D}$ be a efficiently uniformly samplable distribution. There is an efficient protocol $\pi_{\mathsf{Corr},\mathcal{D}}$ that perfectly securely UC-realizes $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ in the $\{\mathcal{F}^n_{\mathsf{SFE},f}\}$-hybrid model against* *environments that (maliciously) corrupt up to t parties. Formally,*

$$\mathcal{F}^{n}_{\mathsf{SFE},f} \overset{\text{perf}}{\rightsquigarrow}_{n} \mathcal{F}^{n}_{\mathsf{Corr},\mathcal{D}} \ . \tag{59}$$

Proof. Let $f \coloneqq f_{\mathcal{D}}$ be the function that ignores the inputs and samples from the distribution $(y_1, ..., y_n) \leftarrow \mathcal{D}$. Each party $\mathsf{P}_i \in P$ inputs `start` into $\mathcal{F}^{n}_{\mathsf{SFE},f_{\mathcal{D}}}$ to obtain and output $y_i = r_i$. When $\mathcal{F}^{n}_{\mathsf{SFE},f}$ is aborted with $(\texttt{abort}, C')$ then each party outputs $(\texttt{abort}, C')$ and terminates. □

FCOT in the Correlated-Randomness model.

Lemma 11 (Correlated-Randomness $\rightsquigarrow_t$ FCOT). *Let $n = n(\lambda)$. Let $\mathcal{D}$ be the SCP-distribution from [30] (the corresponding functionality is $\mathcal{F}^{\text{SCP}}_{\mathsf{Corr}}$). There is an efficient protocol π_{FCOT} that statistically securely UC-realizes $\mathcal{F}^{n}_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^{n}_{\mathsf{Corr},\mathcal{D}}\}$-hybrid model against environments that (maliciously) corrupt up to t parties. Formally,*

$$\mathcal{F}^{n}_{\mathsf{Corr},\mathcal{D}} \overset{\text{stat}}{\rightsquigarrow}_{n} \mathcal{F}^{n}_{\mathsf{FCOT}} \ . \tag{60}$$

Proof. This follows directly from the MPC-completeness of the Correlated-Randomness model in [30]. The high-level idea of the construction in [30] is that the parties use their correlated randomness to commit to their inputs and randomness and then to prove via zero-knowledge that each computation step is consistent with their initial commitments. We refer the reader to [30] for the detailed construction. □

E Trust Graph from Broadcast

Here we provide the full protocol π_{TG} alongside the proof, that π_{TG} really realizes $\mathcal{F}^n_{\mathsf{TG}}$ in the $\{\mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model, which proves the following lemma:

Lemma 6. *Let $n = n(\lambda)$. There is an efficient protocol π_{TG} that perfectly securely UC-realizes $\mathcal{F}^n_{\mathsf{TG}}$ in the $\{\mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against environments that (maliciously) corrupt up to n parties. Formally,*

$$\mathcal{F}^n_{\mathsf{BC}} \overset{\text{perf}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{TG}} \ . \tag{16}$$

Proof. We prove our statement by providing a protocol description for π_{TG} and prove it secure by providing a simulator. Denote the set of parties by $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$. The protocol is given as follows:

Protocol $\mathcal{F}^n_{\mathsf{TG}}$

1. **At the onset** each party $\mathsf{P}_i \in P$ starts with a graph $G_i \coloneqq (P, E_i)$ with $E_i \coloneqq \emptyset$.
2. **On input** $(\texttt{conflict}, \mathsf{P}_i)$ from $\mathcal{Z}$ to P_j, P_j inputs $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_i))$ to $\mathcal{F}^n_{\mathsf{BC}}$.
3. **On output** $(\texttt{output}, \mathsf{P}_j, (\texttt{conflict}, \mathsf{P}_i))$ from $\mathcal{F}^n_{\mathsf{BC}}$ with $i \neq j$, all parties $\mathsf{P} \in P$ add $\{\mathsf{P}_i, \mathsf{P}_j\}$ to E_i.
4. **On input** $(\texttt{query})$ from $\mathcal{Z}$ to P_i, P_i outputs it own G_i.
5. **On output** $(\texttt{abort}, C')$ from $\mathcal{F}^n_{\mathsf{BC}}$ each party outputs $(\texttt{abort}, C')$ and terminates.

A simulator for this protocol is straightforward:

Simulator for $\mathcal{F}^n_{\mathsf{TG}}$

1. **On input** $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_i))$ from a corrupted party P_j to $\mathcal{F}^n_{\mathsf{BC}}$, the simulator ignores the input if $i = j$. Otherwise, $\mathcal{S}$ inputs $(\texttt{conflict}, \mathsf{P}_i)$ into $\mathcal{F}^n_{\mathsf{TG}}$ in the name of P_j. Naturally, the simulator forwards P_j's input $(\texttt{conflict}, \mathsf{P}_i)$ to $\mathcal{F}^n_{\mathsf{BC}}$.
2. **On output** $(\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i)$ from $\mathcal{F}^n_{\mathsf{TG}}$ to $\mathcal{S}$, $\mathcal{S}$ inputs $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ into $\mathcal{F}^n_{\mathsf{BC}}$ in the name of P_j.
3. **On output** $(\texttt{abort}, C')$ from all simulated parties, the simulator inputs $(\texttt{abort}, C')$ into the ideal $\mathcal{F}^n_{\mathsf{COM}}$.

The simulator provides an indistinguishable view for $\mathcal{Z}$. To prove our claim we provide a series of hybrid games that start from an honest execution in the real world and ends with the ideal world where the behavior of honest parties is simulated by a simulator. Intuitively, this is no problem as there are no secret inputs, but more formally:

$\mathcal{H}_1$ This is the real world execution where all honest parties follow the protocol.

$\mathcal{H}_2$ In this game there are two major changes from $\mathcal{H}_1$: We introduce a dummy functionality $\widetilde{\mathcal{F}}^n_{\mathsf{TG}}$ that forwards all inputs to the simulator, and honest parties are replaced by dummy parties that forward their inputs to $\mathcal{F}^n_{\mathsf{TG}}$ and the corresponding simulator executes the code of the respective party. The simulator now also simulates the $\mathcal{F}^n_{\mathsf{BC}}$ setup, i.e., malicious parties interact with the simulated setup.

This change is purely syntactical, the same code gets executed by the simulator as in the real execution.

$\mathcal{H}_3$ Now the dummy functionality is replaced with the ideal functionality $\mathcal{F}^n_{\mathsf{TG}}$ and the simulator is correct simulator given above.

In particular, whenever a corrupted party P_j inputs $(\texttt{conflict}, \mathsf{P}_i)$ into the simulated $\mathcal{F}^n_{\mathsf{BC}}$ the simulator inputs $(\texttt{conflict}, \mathsf{P}_i)$ into $\mathcal{F}^n_{\mathsf{TG}}$ in the name of P_j.

This game is the ideal world execution; it is indistinguishable from the previous game because whenever the simulated honest parties output a new conflict the dummy parties also output that conflict.

Note here that the simulator does not have to manage inputs $(\texttt{query})$ as it involves no interaction. Honest parties merely forward the request and obtain the correct Trust Graph G. And corrupted parties query their local copy. $\square$

F Global Commitment from FCOT

We present a protocol, which realizes $\mathcal{F}^n_{\mathsf{COM}}$ in a $(\mathcal{F}^n_{\mathsf{FCOT}})$-hybrid model, and thus prove the following lemma:

Lemma 12. *Let $n = n(\lambda)$. There is an efficient protocol π_{COM} that statistically securely UC-realizes $\mathcal{F}^n_{\mathsf{COM}}$ in the $\{\mathcal{F}^n_{\mathsf{FCOT}}\}$-hybrid model against environments that (maliciously) corrupt up to t parties. Formally,*

$$\mathcal{F}^n_{\mathsf{FCOT}} \overset{\text{perf}}{\rightsquigarrow}_n \mathcal{F}^n_{\mathsf{COM}} \ . \tag{61}$$

Proof. We denote the parties by $P \coloneqq \{\mathsf{S}, \mathsf{R}_1 = \mathsf{R}, \mathsf{R}_2 = \mathsf{W}_1, ..., \mathsf{R}_{n-1} = \mathsf{W}_{n-2}\}$. We start by sketching the protocol:

Protocol π_{COM}

1. **At the onset** the receiver R inputs $(\texttt{commit}, 0)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$.
2. **On input** $(\texttt{commit}, m)$ for $m \in \{0,1\}^\lambda$ from $\mathcal{Z}$ to S, S samples a uniformly random $m^0 \leftarrow \{0,1\}^\lambda$ and computes $m^1 := m \oplus m^0$. Then S inputs $(\texttt{messages}, m^0, m^1)$ to $\mathcal{F}^n_{\mathsf{FCOT}}$. Any other party $\mathsf{P} \neq \mathsf{S}$ ignores the message.
3. **On input** $(\texttt{receipt transfer})$ from $\mathcal{F}^n_{\mathsf{FCOT}}$, the R_i outputs $(\texttt{receipt commit})$.
4. **On input** $(\texttt{open})$ from $\mathcal{Z}$ to S, S inputs $(\texttt{open message}, 0)$ and $(\texttt{open message}, 1)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$.
5. **On output** $(\texttt{open message}, 0, m^0)$ and $(\texttt{open message}, 1, m^1)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ to any receiver R_i for $i \in [n-1]$, R_i outputs $(\texttt{open}, (m^0 \oplus m^1))$.
6. **On output** $(\texttt{abort}, C')$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ each party outputs $(\texttt{abort}, C')$ and terminates.

The protocol simply uses the committed nature of the messages provided by $\mathcal{F}_{\mathsf{FCOT}}$.

A simulator for this case is straightforward, since all the secrets are sent to the hybrid functionality $\mathcal{F}^n_{\mathsf{FCOT}}$:

Simulator for π_{COM}

1. **On input** $(\texttt{messages}, m^0, m^1)$ from a corrupted sender S to $\mathcal{F}^n_{\mathsf{FCOT}}$, $\mathcal{S}$ reconstructs $m := m^0 \oplus m^1$ and sends $(\texttt{commit}, m)$ to $\mathcal{F}^n_{\mathsf{COM}}$ in the name of S.
2. **On output** $(\texttt{receipt commit})$ from $\mathcal{F}^n_{\mathsf{COM}}$, $\mathcal{S}$ samples two messages $m^0 \overset{\$}{\leftarrow} \{0,1\}^\lambda$ and $m^1 \overset{\$}{\leftarrow} \{0,1\}^\lambda$ uniformly at random and simulates $\mathcal{F}^n_{\mathsf{FCOT}}$ with sender input $(\texttt{messages}, m^0, m^1)$.
3. **On input** $(\texttt{choice}, c)$ for $c \in \{0,1\}$ by a corrupted receiver R_1, $\mathcal{S}$ stores c and reports $(\texttt{receipt transfer})$ to $\mathcal{Z}$ and simulates $\mathcal{F}^n_{\mathsf{FCOT}}$ accordingly.
4. **On output** $(\texttt{open message}, 0, m^0)$ and $(\texttt{open message}, 1, m^1)$ from $\mathcal{F}_{\mathsf{FCOT}}$ to all simulated parties, $\mathcal{S}$ sends $(\texttt{open})$ to $\mathcal{F}^n_{\mathsf{COM}}$ in the name of the sender.
5. **On output** $(\texttt{open}, m)$ from $\mathcal{F}^n_{\mathsf{COM}}$, $\mathcal{S}$ equivocates the commitment by setting $m^{\overline{c}} := m \oplus m^c$ for $\mathcal{F}^n_{\mathsf{FCOT}}$ and outputs $(\texttt{open message}, 0, m^0)$ and $(\texttt{open message}, 1, m^1)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ to all other parties.
6. **On output** $(\texttt{abort}, C')$ from all simulated parties, the simulator inputs $(\texttt{abort}, C')$ into the ideal $\mathcal{F}^n_{\mathsf{COM}}$.

The simulator provides an indistinguishable view:

$\mathcal{H}_1$ This is the real world execution where all honest parties follow the protocol.

$\mathcal{H}_2$ In this game there are two major changes from $\mathcal{H}_1$: We introduce a dummy functionality $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$ that forwards all inputs to the simulator, and honest parties are replaced by dummy parties that forward their inputs to $\mathcal{F}^n_{\mathsf{COM}}$ and the corresponding simulator executes the code of the respective party. Again, this change is purely syntactical.

$\mathcal{H}_3$ In this game we modify the dummy functionality $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$ s.t. when it receives $(\texttt{commit}, m)$ from the dummy sender or the simulator in the name of the corrupted sender it outputs $(\texttt{receipt commit})$ to all other parties.

Moreover, the simulator now recovers $m \leftarrow m^0 \oplus m^1$ from the corrupted sender's input $(\texttt{messages}, m^0, m^1)$ input into $\mathcal{F}^n_{\mathsf{FCOT}}$ and inputs $(\texttt{commit}, m)$ into $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$.

When the simulator receives $(\texttt{receipt commit})$ it let's the simulated sender input $(\texttt{messages}, 0, 0)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$.

When the simulator receives $(\texttt{open}, m)$ from $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$ it equivocates the simulated setup $\mathcal{F}^n_{\mathsf{FCOT}}$ to $(\texttt{open message}, 0, m^0)$ and $(\texttt{open message}, 1, m^1)$ for random $m^0 \leftarrow \{0,1\}$ and $m^1 \leftarrow m \oplus m^0$.

As before these modification do not change the output behavior of the game. For a corrupted sender, the dummy parties output the same value as the simulated honest parties because the simulator extracts $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$'s input from the corrupted sender. For an honest sender, the dummy parties output the same value as the simulated parties because the simulator equivocates the output of the simulated $\mathcal{F}^n_{\mathsf{FCOT}}$.

$\mathcal{H}_4$ In this game the simulator aborts, iff the message extracted from the corrupted sender's local input and the opened message output by the (honest) simulated parties differ.

By definition of $\mathcal{F}^n_{\mathsf{FCOT}}$ this never happens.

$\mathcal{H}_5$ In this game we modify the dummy functionality to be $\mathcal{F}^n_{\mathsf{COM}}$, i.e., when it receives $(\texttt{open})$ from the honest sender or from the simulator in the name of the corrupted sender it outputs $(\texttt{open}, m)$ where m is the previously committed message.

This change is purely syntactical conditioned on the fact that the extracted message (from the corrupted sender) and the opened message by the simulated resp. dummy parties is the same.

$\mathcal{H}_6$ In this game the simulator has no formal abort condition. This is the ideal world execution. Note that the previous abort condition only happened with negligible probability in the first place.

Thus, the claim follows. □

Alternatively, one can also directly use the committed choice bit, removing the need for sharing the message in the inputs for $\mathcal{F}^n_{\mathsf{FCOT}}$ at the cost of only committing to one bit.

References

[1] G. Asharov and C. Orlandi. "Calling Out Cheaters: Covert Security with Public Verifiability". In: *ASIACRYPT 2012*. Ed. by X. Wang and K. Sako. Vol. 7658. LNCS. Springer, Heidelberg, December 2012, pp. 681–698.

[2] Y. Aumann and Y. Lindell. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries". In: *TCC 2007*. Ed. by S. P. Vadhan. Vol. 4392. LNCS. Springer, Heidelberg, February 2007, pp. 137–156.

[3] C. Baum, E. Orsini, and P. Scholl. "Efficient Secure Multiparty Computation with Identifiable Abort". In: *TCC 2016-B, Part I*. Ed. by M. Hirt and A. D. Smith. Vol. 9985. LNCS. Springer, Heidelberg, October 2016, pp. 461–490.

[4] C. Baum et al. “Efficient Constant-Round MPC with Identifiable Abort and Public Verifiability”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, August 2020, pp. 562–592.

[5] D. Beaver. “Multiparty Protocols Tolerating Half Faulty Processors”. In: *CRYPTO’89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, August 1990, pp. 560–572.

[6] M. Ben-Or, R. Canetti, and O. Goldreich. “Asynchronous secure computation”. In: *25th ACM STOC*. ACM Press, May 1993, pp. 52–61.

[7] E. Boyle et al. “Compressing Vector OLE”. In: *ACM CCS 2018*. Ed. by D. Lie et al. ACM Press, October 2018, pp. 896–912.

[8] E. Boyle et al. “Correlated Pseudorandom Functions from Variable-Density LPN”. In: *61st FOCS*. IEEE Computer Society Press, November 2020, pp. 1069–1080.

[9] E. Boyle et al. “Efficient Pseudorandom Correlation Generators: Silent OT Extension and More”. In: *CRYPTO 2019, Part III*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, August 2019, pp. 489–518.

[10] R. Canetti. *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. Cryptology ePrint Archive, Report 2000/067. https://eprint.iacr.org/2000/067. 2000.

[11] R. Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd FOCS*. IEEE Computer Society Press, October 2001, pp. 136–145.

[12] R. Canetti and M. Fischlin. “Universally Composable Commitments”. In: *CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. LNCS. Springer, Heidelberg, August 2001, pp. 19–40.

[13] R. Canetti et al. “Universally Composable Security with Global Setup”. In: *TCC 2007*. Ed. by S. P. Vadhan. Vol. 4392. LNCS. Springer, Heidelberg, February 2007, pp. 61–85.

[14] R. Canetti et al. “Universally composable two-party and multi-party secure computation”. In: *34th ACM STOC*. ACM Press, May 2002, pp. 494–503.

[15] B. Chor and L. Moscovici. “Solvability in Asynchronous Environments (Extended Abstract)”. In: *30th FOCS*. IEEE Computer Society Press, October 1989, pp. 422–427.

[16] R. Cleve. “Limits on the Security of Coin Flips when Half the Processors Are Faulty (Extended Abstract)”. In: *18th ACM STOC*. ACM Press, May 1986, pp. 364–369.

[17] R. Cohen and Y. Lindell. “Fairness versus Guaranteed Output Delivery in Secure Multiparty Computation”. In: *ASIACRYPT 2014, Part II*. Ed. by P. Sarkar and T. Iwata. Vol. 8874. LNCS. Springer, Heidelberg, December 2014, pp. 466–485.

[18] G. Couteau, P. Rindal, and S. Raghuraman. “Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes”. In: *CRYPTO 2021, Part III*. Ed. by T. Malkin and C. Peikert. Vol. 12827. LNCS. Virtual Event: Springer, Heidelberg, August 2021, pp. 502–534.

[19] C. Crépeau. “Efficient Cryptographic Protocols Based on Noisy Channels”. In: *EUROCRYPT’97*. Ed. by W. Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 306–317.

[20] C. Crépeau. “Verifiable Disclosure of Secrets and Applications (Abstract)”. In: *EUROCRYPT’89*. Ed. by J.-J. Quisquater and J. Vandewalle. Vol. 434. LNCS. Springer, Heidelberg, April 1990, pp. 150–154.

[21] C. Crépeau and J. Kilian. “Achieving Oblivious Transfer Using Weakened Security Assumptions (Extended Abstract)”. In: *29th FOCS*. IEEE Computer Society Press, October 1988, pp. 42–52.

[22] C. Crépeau, J. van de Graaf, and A. Tapp. "Committed Oblivious Transfer and Private Multi-Party Computation". In: *CRYPTO'95*. Ed. by D. Coppersmith. Vol. 963. LNCS. Springer, Heidelberg, August 1995, pp. 110–123.

[23] M. Fitzi et al. "Minimal Complete Primitives for Secure Multi-party Computation". In: *CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. LNCS. Springer, Heidelberg, August 2001, pp. 80–100.

[24] R. Gennaro et al. "On 2-Round Secure Multiparty Computation". In: *CRYPTO 2002*. Ed. by M. Yung. Vol. 2442. LNCS. Springer, Heidelberg, August 2002, pp. 178–193.

[25] O. Goldreich, S. Micali, and A. Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *19th ACM STOC*. Ed. by A. Aho. ACM Press, May 1987, pp. 218–229.

[26] S. Goldwasser and S. Micali. "Probabilistic Encryption". In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299.

[27] V. Goyal et al. "Founding Cryptography on Tamper-Proof Hardware Tokens". In: *TCC 2010*. Ed. by D. Micciancio. Vol. 5978. LNCS. Springer, Heidelberg, February 2010, pp. 308–326.

[28] Y. Ishai, E. Kushilevitz, and A. Paskin. "Secure Multiparty Computation with Minimal Interaction". In: *CRYPTO 2010*. Ed. by T. Rabin. Vol. 6223. LNCS. Springer, Heidelberg, August 2010, pp. 577–594.

[29] Y. Ishai, R. Ostrovsky, and H. Seyalioglu. "Identifying Cheaters without an Honest Majority". In: *TCC 2012*. Ed. by R. Cramer. Vol. 7194. LNCS. Springer, Heidelberg, March 2012, pp. 21–38.

[30] Y. Ishai, R. Ostrovsky, and V. Zikas. "Secure Multi-Party Computation with Identifiable Abort". In: *CRYPTO 2014, Part II*. Ed. by J. A. Garay and R. Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, August 2014, pp. 369–386.

[31] Y. Ishai, M. Prabhakaran, and A. Sahai. "Founding Cryptography on Oblivious Transfer - Efficiently". In: *CRYPTO 2008*. Ed. by D. Wagner. Vol. 5157. LNCS. Springer, Heidelberg, August 2008, pp. 572–591.

[32] Y. Ishai et al. "Zero-knowledge from secure multiparty computation". In: *39th ACM STOC*. Ed. by D. S. Johnson and U. Feige. ACM Press, June 2007, pp. 21–30.

[33] J. Katz et al. "Universally Composable Synchronous Computation". In: *TCC 2013*. Ed. by A. Sahai. Vol. 7785. LNCS. Springer, Heidelberg, March 2013, pp. 477–498.

[34] J. Kilian. "Founding Cryptography on Oblivious Transfer". In: *20th ACM STOC*. ACM Press, May 1988, pp. 20–31.

[35] Y. Lindell, E. Oxman, and B. Pinkas. "The IPS Compiler: Optimizations, Variants and Concrete Efficiency". In: *CRYPTO 2011*. Ed. by P. Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, August 2011, pp. 259–276.

[36] C. Orlandi, P. Scholl, and S. Yakoubov. "The Rise of Paillier: Homomorphic Secret Sharing and Public-Key Silent OT". In: *EUROCRYPT 2021, Part I*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12696. LNCS. Springer, Heidelberg, October 2021, pp. 678–708.

[37] R. Pass. "On Deniability in the Common Reference String and Random Oracle Model". In: *CRYPTO 2003*. Ed. by D. Boneh. Vol. 2729. LNCS. Springer, Heidelberg, August 2003, pp. 316–337.

[38] T. Rabin and M. Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract)". In: *21st ACM STOC*. ACM Press, May 1989, pp. 73–85.

[39] A.-R. Sadeghi, T. Schneider, and M. Winandy. “Token-Based Cloud Computing”. In: *Trust and Trustworthy Computing*. Ed. by A. Acquisti, S. W. Smith, and A.-R. Sadeghi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 417–429.

[40] A. Shamir. “How to Share a Secret”. In: *Communications of the Association for Computing Machinery* 22.11 (November 1979), pp. 612–613.

[41] M. Simkin, L. Siniscalchi, and S. Yakoubov. “On Sufficient Oracles for Secure Computation with Identifiable Abort”. In: *Security and Cryptography for Networks*. Ed. by C. Galdi and S. Jarecki. Cham: Springer International Publishing, 2022, pp. 494–515.

[42] J. Wan et al. “Expected Constant Round Byzantine Broadcast Under Dishonest Majority”. In: *TCC 2020, Part I*. Ed. by R. Pass and K. Pietrzak. Vol. 12550. LNCS. Springer, Heidelberg, November 2020, pp. 381–411.

[43] S. Wolf and J. Wullschleger. “Oblivious Transfer Is Symmetric”. In: *EUROCRYPT 2006*. Ed. by S. Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 222–232.