# Constructing Secure Multi-Party Computation with Identifiable Abort

**Cheater Identification via Graph Theory in the Dishonest Majority Setting**

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

[1] ETH Zurich, Zürich, Switzerland, nicholas.brandt@inf.ethz.ch
[2] Karlsruhe Institute of Technology, Germany, {sven.maier2,joern.mueller-quade}@kit.edu
[3] FZI Research Center for Information Technology, Germany, tobias.mueller@fzi.de

**Abstract.** We propose an intuitive approach for constructing and analyzing Multi-Party Computation protocols with Identifiable Abort (ID-MPC) based on simple graph-theory.
On a high level, in our approach, honest parties publicly announce conflicts with malicious parties via broadcast whenever they catch them misbehaving, thus inducing a *Conflict Graph* (CG). We directly link the sufficient and necessary conditions for the (identifiable) abort of a protocol to publicly verifiable graph-theoretical properties of the Conflict Graph.
To demonstrate its power, we use our technique to reduce the necessary requirements for ID-MPC in the Universal Composability framework with a dishonest majority. State-of-the-art protocols in the dishonest majority setting are posited in the Correlated-Randomness model where one $n$-party setup provides randomness that is $n$-wise correlated to all other parties' randomness. Using our technique we are able to reduce the degree of correlation in the this randomness from $n$ to $n-1$.
Additionally, if $n$ is sufficiently small, then our upper bound can be transitively expanded, i.e., for $t \leq n-3$ corruptions among $n$ parties we can construct $n$-party ID-MPC from correlated randomness among each set of $t+2$ parties.

**Keywords:** Multi-Party Computation · Identifiable Abort · Conflict Graph · Dishonest Majority · Universal Composability

## Contents



## 1 Introduction

Secure Multi-Party Computation (MPC) has been studied intensely since its conception in the 1980's. For the honest-majority case, i.e. where a strict majority of parties is honest, the fundamental (in-)feasibility results has been established fairly early [29, 7, 44, 5]. In contrast, the dishonest-majority case suffers from an impossibility of fairness [21]. To sidestep this impossibility

protocols in the dishonest-majority setting [40, 37] have settled for security with abort where the adversary can abort the protocol at any time. While this notion ensures that protocols are secure and private, it opens the door for Denial-of-Service attacks. This undesirable property has been remedied by the introduction of security with Identifiable Abort (IA) (formally in [36] and before in [29, 33, 2, 35] among others) where—upon abort—the identity of a malicious party is revealed. As such, this notion of security is desirable because it acts a deterrent against cheating if coupled with some form of penalty mechanism.
The state-of-the-art MPC protocols with Identifiable Abort (ID-MPC) are posited in the Correlated-Randomness model [36] where a setup functionality provides all parties with $n$-wise correlated randomness. In light of the insufficiency of Oblivious Transfer (OT), i.e., pairwise correlations, from [35] the subsequent work [36] states the following:

> *"Indeed, the insufficiency of OT [shown in [35]] implies that pairwise correlated randomness is not sufficient for information-theoretic ID-MPC, but leaves open the question of whether or not n-wise correlations are, which is answered affirmatively here."*

In this work we take this question further and ask whether $k^*$-wise correlations are sufficient for $k^* < n$ which we also answer affirmatively (given broadcast) for $k^* \leq t + 2 \leq n - 1$ where $t$ is the max. number of corrupted parties; see Fig. 1 for an overview. For $n/2 \leq t \leq n - 3$ we present the first upper bound on the minimal setup size that suffices for UC-secure $n$-party computation with Identifiable Abort.

Aside from furthering the theoretical understanding of the minimal assumptions for statistically secure ID-MPC, our result also weakens the assumptions for practical (computationally secure) ID-MPC. In particular, the computationally secure ID-MPC protocol for any adaptively secure OT protocol in [36] requires a Common Reference String (CRS) setup for *all* parties. Combining this protocol with our construction reduces the size, i.e., the number of parties, of the CRS from $n$ to $n - 1$ parties, if at least three parties are honest.
Since our works shows that a CRS for all parties is not necessary, this poses the question of the minimal "locality", e.g. CRS size, that is necessary for ID-MPC. In this work, we make first steps towards answering this question.

### 1.1 Contribution

Our three contributions are:

***Identification via conflicts.*** Our main contribution is a formalization of an intuitive mechanism for cheater identification based on conflicts between parties. The basic idea is that honest parties publicly announce conflicts with parties that are "caught cheating". Intuitively related notions have been used in various works [38, 34, 36, 3, 4, 48], often as tools in proofs. However, to our knowledge, they were never suitably formalized and comprehensively studied. Through our formalization of the *Conflict Graph* (CG) approach we can closely link the conditions for Identifiable Abort of a ID-MPC protocol to two publicly verifiable graph-theoretical properties of the CG.

Technically, we define a special setup functionality $\mathcal{F}_{\mathsf{CG}}^n$ where any party $\mathsf{P} \in P$ can register their conflicts, thus adding the corresponding edges to the Conflict Graph $G = (P, E)$ which is returned by $\mathcal{F}_{\mathsf{CG}}^n$ upon query by any party.
Moreover, we define a specific way of using the functionality $\mathcal{F}_{\mathsf{CG}}^n$ which we call *abort-respecting* and show that any protocol can be augmented to use $\mathcal{F}_{\mathsf{CG}}^n$ in an abort-respecting way s.t. the protocol retains its functionality and security. This allows us to lift statements for abort-respecting protocols to all protocols. That is, the impossibility of an 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.

Finally, we show that $\mathcal{F}_{\mathsf{CG}}^n$ can be realized using only broadcast. This is due to the fact that $\mathcal{F}_{\mathsf{CG}}^n$ merely stores all broadcasted conflicts in a concise way.

***Committed Oblivious Transfer is complete for ID-MPC.*** In their full version Ishai, Ostrovsky, and Seyalioglu [35] show that Oblivious Transfer (OT) is not complete for statistically secure ID-MPC in the stand-alone model. We thus reformulate Crépeau's *Committed* OT [24, 26] as an

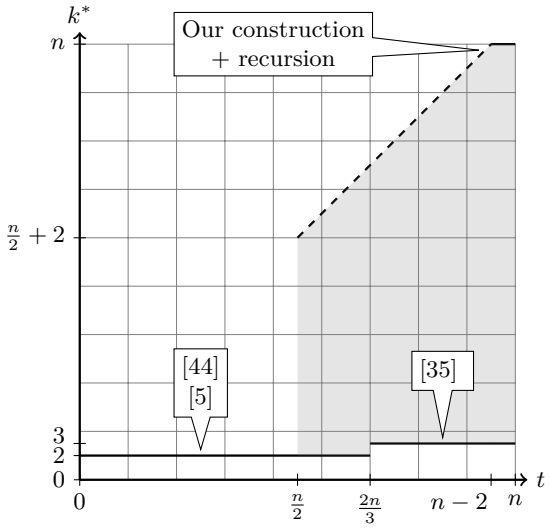



**Fig. 1:** Upper bound of the minimal complete cardinality $k^*$ for statistically UC-secure ID-MPC vs. *maximal* number of malicious parties $t$ given broadcast. The grey area represents the possible region of $k^*$. The dashed lines indicate our bounds ($n/2$ recursions for $n \in \mathcal{O}(\ln \lambda / \ln \ln \lambda)$).

ideal functionality $\mathcal{F}_{\mathsf{FCOT}}^n$ and prove it is complete for ID-MPC. The Fully Committed Oblivious Transfer (FCOT) lets all $n$ parties obtain a receipt after the OT has been performed: a sender and a receiver have secret inputs as in the classical OT, the remaining $(n-2)$ *witnesses* do not have any input. After the OT-phase, both the sender and the receiver are committed to their inputs independently and can open them at a later point to all other parties; even after the receiver obtained only one message, the sender can open both $m_0$ *and* $m_1$, and the receiver can open the choice bit $c$ such that no party can lie about their actual input.

The main idea behind the use of the committed OT to obtain Identifiable Abort is the observation that in the IPS-compiler [37] the abort occurs whenever an honest party notices some misbehavior on some server. Though, the honest party cannot (in the multi-party case) point with confidence to the actual cheater. However, if all OTs were committed and witnessed by all other parties, then the honest party that noticed the cheating can simply challenge all parties to open all previous communication and thus all parties can retrace the computation and therefore identify the cheater or the falsely accusing party.

***Expanding ID-MPC.*** Finally, to demonstrate the power of our technique we provide an expansion from $(n-1)$-party ID-MPC and $n$-party broadcast to $n$-party ID-MPC. This implies an upper bound for the minimal complete cardinality[1] for $n$-party Secure Function Evaluation (SFE) in the style of [28], assuming that at least three parties are honest.

More precisely, we give a protocol that expands a commitment from size $n-1$ to $n$. In a second step, we expand FCOT from $n-1$ to $n$ parties using the previously global commitment. Lastly, we observe that FCOT, SFE and Correlated-Randomness imply each other information-theoretically.

Additionally, if the number of parties is sufficiently small, e.g. $n \in \mathcal{O}(\ln \lambda / \ln \ln \lambda)$, then we can apply our result recursively. For $t \leq n-3$ corruptions we can construct $n$-party MPC from correlated randomness among each set of $t+2$ parties (compare Fig. 1).

We want to stress that the condition $t \leq n-3$ is not necessary for using the CG in general. In particular, the connection between the abort of a protocol and the graph properties of the CG hold for any $n \geq 1$ and $0 \leq t \leq n$.

***A note on efficiency.*** As we view this work as mainly a feasibility result we do not consider (practical) efficiency of our protocol. However, we can make some high-level remarks.
First, our construction only uses information-theoretic tools, e.g. secret-sharing and some efficient

[1] The minimal size of setups that suffice for general ID-MPC with $n$ parties.

graph-operations on the Conflict Graph. As such our protocol does not require heavy computationally.

On the other hand, the communication complexity—often the practically relevant parameter—is quite high. The abort-respecting property which is at the heart of our technique requires that parties gradually declare conflicts until *biseparation* (see Definition 6) is reached on some subgraph or the overall Conflict Graph. This gradual declaration of conflicts requires $\Theta(n)$ rounds in the worst-case. Moreover, the adversary can abort up to $\mathcal{O}(n)$ setups for $t \leq n-3$. With the obvious approach of "abort-then-biseparate" this already gives a round complexity of $\mathcal{O}(n^2)$. This problem also seems to arise in the concurrent work [48] 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.

Lastly, we waht to mention that replacing *one* setup with *many* slightly smaller setups seems like a bad tradeoff. However, all these smaller setups can be performed in the same rounds because they are independent of each other. Thus, the overall communication complexity might increase proportional to the number of setups but the number of rounds increase only additively.

### 1.2 Related and concurrent work

In Table 1 we give an overview of the related work regarding the theoretical (in)-feasibility of statistically secure ID-MPC. As mentioned in Section 1 Ishai, Ostrovsky, and Seyalioglu [35] showed that Oblivious Transfer is insufficient for statistically secure ID-MPC. This eventually lead to [36] showing that $n$-wise correlated randomness is sufficient.

Concurrently to this work, Simkin, Siniscalchi, and Yakoubov [48] give a marginal improvement over our result, although they present it in the stand-alone model while our result is universally composable. Their upper bound $k^* \leq t+1 \leq n-1$ is marginally better than our bound $k^* \leq t+2 \leq n-1$. However, both works only support polynomially many parties $n \in \text{poly}(\lambda)$ for a constant expansion $\ell := n-t \in \Theta(1)$. For larger expansions by recursive application of the expansion protocols the supported number of parties drops rapidly since the overall runtime grows exponentially in the number of recursions.
Their approach uses an new form of identifiable secret-sharing with public and private shares. There, one party $\mathsf{P}$ 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 $\mathsf{P}$ who reconstructs its randomness. If reconstruction fails to due faulty shares sent by malicious parties, then party $\mathsf{P}$ 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.
Here, we find an example of usage of conflicts in an *adhoc manner*. In contrast, we position our work as furthering the fundamental understanding of the power and limitations of this approach in general, and give our construction only as a byproduct of this more general understanding.

Moreover, in the approach of [48] it is not clear how one would adapt their technique to setups of size $n-2$ and smaller.
In contrast, subsequently and independently to the online version of this work Brandt [15]—based on our technique—presented stronger graph-theoretical lemmas that result in *tight bounds* for the setup size. In particular, he shows that setups of size $k^* := \min(n, \lfloor (n-1)/(n-t) \rfloor + \lfloor n/(n-t) \rfloor - 1)$ are sufficient while setups of size $k^*-1$ are not—if a broadcast is available. We interpret this result as a testament to the power of our formalization of Conflict Graphs. Furthermore, his result suffers less from the severe restrictions on the number of parties because his approach does not need recursive application of the expansion protocol.

At this point we want to relativize the importance of our result in practice. Since information-theoretic ID-MPC without any setup is impossible in the dishonest-majority setting research has been focused on finding a *minimal*[2] setup which is efficiently realizable via cryptographic assumptions, trusted hardware [31, 46] or noisy channels [25, 23]. From a practical point of view the computationally secure generation of correlated randomness from any adaptively secure OT protocol [36] might be satisfying (compare Table 1). Indeed, many works have followed this approach

[2] Not necessarily minimal in the number of parties but rather minimal in a practically implementable way.

| Reference | Security | Model | Result | Technique |
|---|---|---|---|---|
| [35] | stat | stand-alone | $\mathcal{F}^2$ is insufficient for $t \geq 2n/3$ | Secret-Sharing |
| [36] | stat | UC, stand-alone | $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ is complete | Setup+Commit+Prove |
| [36] | comp | UC, stand-alone | $\pi^{\mathsf{OT}} + \mathcal{F}^n_{\mathsf{CRS}} \rightsquigarrow \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ | Setup+Commit+Prove |
| [48] | stat | stand-alone | $\mathcal{F}^{n-1}_{\mathsf{Corr},\mathcal{D}} \rightsquigarrow \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ for $t \leq n-2$ | Secret-Sharing |
| This work | stat | UC | $\mathcal{F}^{n-1}_{\mathsf{Corr},\mathcal{D}} \rightsquigarrow \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ for $t \leq n-3$ | *Conflict Graph* |
| [15] | stat | UC | $\mathcal{F}^{k^*}_{\mathsf{Corr},\mathcal{D}} \rightsquigarrow \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ | *Conflict Graph* |
| [15] | comp | UC | $\mathcal{F}^{k^*-1}_{\mathsf{Corr},\mathcal{D}} \not\rightsquigarrow \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ | *Conflict Graph* |

**Table 1:** Overview of related work on the foundations of Multi-Party Computation with Identifiable Abort in the dishonest majority setting with broadcast. $\pi^{\mathsf{OT}}$ is any adaptively secure Oblivious Transfer (OT) protocol, $\mathcal{F}^n_{\mathsf{CRS}}$ is the Common Reference String (CRS) functionality from [18], and $t$ is the max. number of corrupted parties. Subsequently and independently to our work [15] provided stronger graph-theoretical lemmas for our approach where $k^* := \min(n, \lfloor (n-1)/(n-t) \rfloor + \lfloor n/(n-t) \rfloor - 1)$. Note that the impossibility of [15] does not contradict the computational construction of ID-MPC from any adaptively secure OT protocol and a CRS in [36] because [15] does not assume a CRS of size $n$.

to improve it [32, 8, 9, 14, 13, 11, 12, 43].
Adding to this list, our work explores the theoretical boundaries of generating correlated randomness. Since our techniques can be readily plugged into any construction of correlated randomness for $n$ parties as briefly mentioned in Section 1, we deem it worthwhile to investigate these theoretical limits.

Lastly, we want to mention a curious connection to topology-hiding computation (THC) [42]. There, the topology of the communication network (secret) might roughly be interpreted as the complement graph $\overline{G}$ of the Conflict Graph (public). The impossibility result of topology-hiding broadcast against fail-stop adversaries in [42] is based on the fact that the adversary can disconnect the communication graph—thus cutting off the sender from some receiver. In our model this translates into the statement that the adversary can biseparate (see Definition 6) the overall Conflict Graph. However, in contrast to our setting, in the setting of THC the Conflict Graph is *not* public knowledge and as such the honest parties cannot identify the malicious (blocking) party.
This unexpected link suggests that our graph-theoretical tools might find fruitful application in other fields such as THC. Though, we leave this to future work.

### 1.3 Setting

Our constructions enjoy **statistical security**, also called information-theoretic security, no computational assumptions are made. We only assume the existence of hybrid functionalities or *setups*. This leaves the means of the realization of these setups up to the user, e.g. via physical means such as trusted hardware [31, 46] or noisy channels [25, 23], or again from computational assumptions of choice ([1, 10, 45] to name only a few). We don't explicitly assume additional pairwise secure channels, since they can be emulated by setups of size $\geq 2$.

We focus on **static corruptions** of an arbitrary number of parties. We denote the maximal number of malicious parties by $t < n$.

We assume that all messages sent between parties and ideal functionalities are **authenticated**. We further assume that all parties have access to an $n$-party *broadcast*, which we model as ideal functionality $\mathcal{F}^n_{\mathsf{BC}}$. This broadcast is mainly used to realize our functionality $\mathcal{F}^n_{\mathsf{CG}}$; for more details we refer to Appendix A.1.

We generally assume that the simulator gets notified whenever any party passes input to any functionality. The simulator doesn't learn anything regarding the parties secret inputs (except its length when applicable). It only learns that input was provided.

***The UC Framework.*** We perform our analysis in the Universal Composability (UC) framework [16, 17], which is a strong version of simulation-based security [30, 29]. 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 $\pi$, 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.

The UC framework provides much stronger security guarantees than the standalone model, but comes with some restrictions; without a trusted setup, no protocol $\pi$ can UC-realize functionalities such as Commitments [18], 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.

We assume a **synchronous** communication network, as our Conflict Graph requires that any conflict announced by a party $\mathsf{P}$ will be received by all other parties. In an asynchronous model, the adversary could drop all messages [20, 6], resulting in a situation similar to (anonymous) abort, thus rendering Identifiable Abort essentially useless. In the synchronous model, however, the adversary can only either let the functionality terminate, or abort at the cost of revealing the identity of at least one malicious party. On an intuitive level, we adapt the view from the 2020 version[3] of [16], which describes how synchronous communication can be achieved by using the functionality $\mathcal{F}_{\mathsf{SYN}}$. However, for the sake of succinct protocol descriptions, we omit the usage of $\mathcal{F}_{\mathsf{SYN}}$ in our analysis and just assume that parties are activated in rounds.

We don't assume direct party-party communication. In our hybrid protocol, honest parties have an authenticated connection with the hybrid functionalities; the adversary cannot manipulate messages from honest parties to hybrid functionalities and vice versa. This model corresponds to an adversary that might listen on the network but cannot fabricate false messages from honest parties.

***Identifiable Abort.*** Unfortunately, fairness and thus guaranteed output is impossible against a dishonest majority [21]. On the other hand, the much weaker notion of security with (anonymous) abort, where the adversary can abort the protocol at any time, leaves the protocol vulnerable to Denial-of-Service attacks.

To sidestep this issue we consider in the setting of IA as formalized by Ishai, Ostrovsky, and Zikas [36]. 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.

We use the following notation to clarify our Identifiable Abort property[4]:

**Notation 1 (Functionalities with IA).** *We denote by $\mathcal{F}^n$ an $n$-party functionality with Identifiable Abort.*

**Definition 1 (Identifiable Abort).** *Let $\mathcal{F}^n$ be an ideal $n$-party functionality with parties $P$ and malicious subset $C \subseteq P$. $\mathcal{F}^n$ has **Multi-Identifiable Abort**, iff all (honest) parties yield output $(\texttt{abort}, C')$ when the adversary sends $(\texttt{abort}, C')$ to $\mathcal{F}^n$. If $C' \not\subseteq C$ or $C' = \emptyset$, the message is ignored. $\mathcal{F}^n$ has **Uni-Identifiable Abort**, iff $\mathcal{F}^n$ has* Multi-IA *and $|C'| = 1$.*

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.

[3] Eprint-Version 20200212:021048

[4] The original work [36] uses the notation $\mathcal{F}_{\perp}^{\mathsf{ID}}$.

## 2 Preliminaries

### 2.1 Basics

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

**Notation 2 (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 3 (Subsets).** *For any set $V$ and $k \in \mathbb{N}$ we denote by $\binom{V}{k} \coloneqq \{V' \subseteq V \mid |V'| = k\}$ the set of subsets of cardinality $k$.*

**Notation 4 (Union of disjoint sets).** *Let $V$ and $V'$ be two disjoint sets, i.e. $V \cap V' = \emptyset$. As a help for the reader we emphasize their disjointness in the union operation as $V \uplus V' = V \cup V'$.*

In particular, we use the fact that $|V \uplus V'| = |V| + |V'|$.

**Notation 5 (Realizing ideal functionalities).** *Let $F$ be a set of ideal functionalities (or protocols) $F$, let $\pi^F$ be any protocol in the $F$-hybrid model, and let $\mathcal{F}$ be an ideal functionality. Iff $\pi^F$ securely UC-realizes $\mathcal{F}^n$, we write*

$$\begin{aligned} \pi^F \geq \mathcal{F} \ , \qquad & \textit{or equvalently} \\ \exists \mathcal{S} \, \forall \mathcal{Z} : \mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}(\mathcal{Z}) \approx \mathsf{IDEAL}_{\mathcal{F}, \mathcal{S}}(\mathcal{Z}) & \end{aligned} \tag{1}$$

*where $\mathsf{REAL}_{\pi^F, \mathcal{A}_\mathcal{D}}(\mathcal{Z})$ is the environment's output when running the (hybrid) protocol $\pi^F$ with the dummy adversary[5] and $\mathsf{IDEAL}_{\mathcal{F}, \mathcal{S}}(\mathcal{Z})$ is the environment's output when running a simulated protocol with the simulator $\mathcal{S}$.*

**Notation 6 (Protocol construction).** *Let $F$ be a set of ideal functionalities (or protocols) $F$ and let $\mathcal{F}$ be an ideal functionality. We write $F \rightsquigarrow \mathcal{F}$, iff there is a protocol $\pi^F$ that securely UC-realizes $\mathcal{F}$ in the $F$-hybrid model. More formally:*

$$F \rightsquigarrow \mathcal{F} \iff \exists \pi^F : \pi^F \geq \mathcal{F} \ . \tag{2}$$

*Conversely, we write*

$$F \not\rightsquigarrow \mathcal{F} \iff \forall \pi^F : \pi^F \ngeq \mathcal{F} \ . \tag{3}$$

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

**Notation 7 (Minimal complete cardinality; generalized from [28]).** *For any number of parties $n \in \mathbb{N}_{\geq 2}$ let $k \in \mathbb{N}$ be the smallest number such that $\forall \mathcal{F}^n \, \exists \mathcal{F}^k : \{\mathcal{F}^k\} \cup F \rightsquigarrow \mathcal{F}^n$, then $k$ is the* minimal complete cardinality *for $n$-party MPC relative to some set of ideal setup functionalities $F$.*

The original work [28] defined the minimal complete cardinality relative to $F = \emptyset$ while we consider $F = \{\mathcal{F}_{\mathsf{BC}}^n\}$, i.e., the broadcast functionality defined in Section 2.2.

**Notation 8 (Lists).** *For any list $\gamma \coloneqq (\gamma_i)_{i \in [l]}$ of length $l \in \mathbb{N}$ we denote the list at index set $\nu \subseteq [l]$ by $\gamma_\nu \coloneqq (\gamma_i)_{i \in \nu}$.*

**Definition 2 (Threshold secret sharing).** *For any $\ell_1, \ell_2 \in \mathbb{N} : \ell_1 \leq \ell_2$ an $(\ell_1, \ell_2)$-threshold secret sharing scheme for message space $M$ is defined by a probabilistic algorithm $\text{Share}_{\ell_1, \ell_2}$ and a deterministic algorithm $\text{Recover}_{\ell_1, \ell_2}$ with syntax*

- $\text{Share}_{\ell_1, \ell_2} : M \to (\{0,1\}^\lambda)^{\ell_2} : m \mapsto \mu = (\mu_\kappa)_{\kappa \in [\ell_2]}$
  $\text{Share}_{\ell_1, \ell_2}$ *takes a message $m \in M$ and outputs $\ell_2$ shares such that $\text{Recover}_{\ell_1, \ell_2}$ 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 $\text{Share}_{\ell_1, \ell_2}$, for all messages $m, m' \in M$ and for all sets of indices $\nu \in \binom{[\ell_2]}{\ell_1 - 1}$ it should hold that*

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

[5] Here, $\mathcal{A}_\mathcal{D}$ is the canonical dummy adversary who simply forwards communication from and to $\mathcal{Z}$ (see Section 4.3.1 in [16]).

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

$$\Pr_{\mu \leftarrow \text{Share}_{\ell_1,\ell_2}(m)}\left[\forall \mu' \text{ s.t. } \mu_\nu = \mu'_\nu : \begin{array}{l} \mu = \mu' \iff \text{Recover}_{\ell_1,\ell_2}(\mu') = m \\ \mu \neq \mu' \iff \text{Recover}_{\ell_1,\ell_2}(\mu') = \bot \end{array}\right] \in \text{owhl}(\lambda) \ .$$

For example, we could use Shamir's secret sharing [47] with $\ell_2 = 2\ell_1$. The purpose of these threshold sharing is to ensure that parties input the same shares across multiple setups. To this end, we use the following lemma which bounds the probability that a sufficiently manipulated sharing stays undetected.

**Lemma 1 (Error-detection).** *Let $u, w, s \in \mathbb{N}_{\geq 1}$ s.t. $s, w \leq u$ and let $W \in \binom{[u]}{w}$.*

$$\Pr_{S \leftarrow \binom{[u]}{s}}[W \cap S = \emptyset] = \prod_{i=0}^{s-1} \frac{u-w-i}{u-i} \leq \prod_{i=0}^{s-1} \frac{u-w}{u} \leq (1-w/u)^s \leq 2^{-sw/u} \tag{4}$$

**Notation 9 (Complement graph).** *For any undirected irreflexive graph $G = (V, E)$ we use the notation $\overline{G} \coloneqq (V, \overline{E})$ for the undirected, reflexive* complement *graph with $\overline{E} \coloneqq \binom{V}{2} \setminus E$.*

### 2.2 Functionalities / Setups

In this section, we introduce the ideal functionalities we use. We note that all following functionalities exhibit Identifiable Abort.

Without writing it out explicitly each time, we generally assume that all of our functionalities are inherently *unfair*: public outputs are first sent to the corrupted parties, i.e. the adversary. The adversary can then decide if the functionality should be aborted, thus preventing the honest parties from receiving the output.

***Global functionalities.*** For our constructions, we use the one-to-many commitment from [19, 36] adapted to Identifiable Abort. 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 $\lambda$, parties $P = \{\mathsf{S}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$, 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 $\mathsf{S}$, send $(\texttt{receipt commit})$ to all parties. Ignore further messages $(\texttt{commit}, \cdot)$ from $\mathsf{S}$.
- When receiving $(\texttt{open})$ from party $\mathsf{S}$, send $(\texttt{open}, m)$ to all parties and terminate.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.

Beyond the guarantees that the global commitment inherited from two-party commitments, namely the *binding* and *hiding* properties, $\mathcal{F}^n_{\mathsf{COM}}$ additionally ensures *consistency* in that the committer $\mathsf{C}$ is committed to the *same* message against all receivers. We also make use of the ideal broadcast functionality from [19], with modifications to support Identifiable Abort.

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

$\mathcal{F}_{\mathsf{BC}}^n$ proceeds as follows, running with security parameter $\lambda$, 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 $\mathsf{P}_i$, 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.

In the context of ID-MPC with UC-security we consider $\mathcal{F}_{\mathsf{BC}}^n$ to be a relatively weak functionality; we use it primarily to realize the Conflict Graph functionality $\mathcal{F}_{\mathsf{CG}}^n$ defined in Section 3. We provide an instantiation of $\mathcal{F}_{\mathsf{CG}}^n$ using $\mathcal{F}_{\mathsf{BC}}^n$ in Appendix A.1, thus proving that $\mathcal{F}_{\mathsf{CG}}^n$ too is a relatively weak setup. In particular, the constructions of [36] also require a broadcast.
Indeed, for our formalization of the Conflict Graph the broadcast is actually necessary, since the formal object of the CG is not well-defined if there is no consistent view of conflicts shared by all parties. However, we believe that the concept of identification via conflicts may also be fruitful in the setting were *no broadcast* is present. Here, one would instead consider a *local* CG for each party. We leave the adaptation of our technique to this setting to future work.

We stress that $\mathcal{F}_{\mathsf{BC}}^n$ too can be aborted by the adversary. While intuitively, this breaks our guarantees, as any accusations of honest parties against malicious parties can be suppressed, the reason this does not violate our results comes from the way we use the broadcast and hence the Conflict Graph; while the remaining functionalities usually omit one party during our constructions, the broadcast is only executed over the entire set of parties and hence an abort of a broadcast allows the honest party to identify (at least) one malicious party.

Furthermore, note that it is obvious that a global commitment implies broadcast, by letting the sender $\mathsf{S}$ *commit* and immediately *open* the message.

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

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

$\mathcal{F}_{\mathsf{SFE},f}^n$ proceeds as follows, running with security parameter $\lambda$, 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 $\mathsf{P}_i$. Messages not covered here are ignored.

- When receiving $(\texttt{input}, x_i)$ from $\mathsf{P}_i$ with $x_i \in \{0,1\}^\lambda$, store $(i, x_i)$ and send $(\texttt{receipt}, \mathsf{P}_i)$ to each party $\mathsf{P}_j$ and to $\mathcal{A}$. Ignore further messages from $\mathsf{P}_i$.
- When there are $(i, x_i)$ stored for all $i \in [n]$, then send $(\texttt{output}, y_j)$ to each party $\mathsf{P}_j$ and $(\texttt{output})$ to $\mathcal{A}$, then terminate.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties, and then terminate.

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

Functionality $\mathcal{F}_{\mathsf{Corr},\mathcal{D}}^n$ adapted from [36]

$\mathcal{F}_{\mathsf{Corr},\mathcal{D}}^n$ proceeds as follows, running with security parameter $\lambda$, 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 $\mathsf{P}$ or $\mathcal{S}$, sample $(r_1, ..., r_n) \leftarrow \mathcal{D}$, output $r_i$ to $\mathsf{P}_i$, and terminate.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties, and then terminate.

Next, we formulate a variant of OT introduced by Crépeau [24] under the name of Verifiable OT, which was later described as Committed Oblivious Transfer [26]. We call our formalization as an ideal functionality Fully Committed Oblivious Transfer (FCOT), which extends *normal* OTs in two ways: 1. it includes $n-2$ witnesses, which obtain a *receipt* if the message has been transferred successfully, 2. the sender $\mathsf{S}$ is committed to both messages $m_0$ and $m_1$, and 3. the receiver $\mathsf{R}$ is committed to $c$.

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

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

- When receiving $(\texttt{messages}, m^0, m^1 \in \{0,1\}^\lambda)$ from $\mathsf{S}$, store $m^0, m^1$. Ignore further messages of the type $(\texttt{messages}, \cdot, \cdot)$ from $\mathsf{S}$.
- When receiving $(\texttt{choice}, c \in \{0,1\})$ from $\mathsf{R}$ with $c \in \{0,1\}$, store $c$. Ignore further messages of the type $(\texttt{choice}, \cdot)$ from $\mathsf{R}$.
- When both $m^0, m^1$ and $c$ are stored, send $(\texttt{output}, m^c)$ to $\mathsf{R}$, and $(\texttt{receipt transfer})$ to all other parties and $\mathcal{S}$.
- When receiving $(\texttt{open message}, b \in \{0,1\})$ from $\mathsf{S}$ and $m^0, m^1$ are stored, send $(\texttt{open message}, b, m^b)$ and to all parties and $\mathcal{S}$. Ignore further messages $(\texttt{open message}, b)$ from $\mathsf{S}$.
- When receiving $(\texttt{open choice})$ from $\mathsf{R}$ and $c$ is stored, send $(\texttt{open choice}, c)$ and to all parties and $\mathcal{S}$. Ignore further messages from $\mathsf{R}$.
- When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.

This $n$-party extension of OT is motivated the incompleteness of any pairwise functionality, e.g. OT, for Identifiable Abort as shown by Ishai, Ostrovsky, and Seyalioglu [35]. In particular, the results of Kilian [40] and Ishai, Prabhakaran, and Sahai [37] do not have IA, they only provide security with abort. In the construction of [37], any party $\mathsf{P}$ notices when another party $\mathsf{P}'$ misbehaves towards $\mathsf{P}$, then the protocol can abort. However, a third party $\mathsf{P}^*$ need not necessarily notice such misbehavior of $\mathsf{P}'$ towards $\mathsf{P}$. The intuitive fix is to replace all calls to the OT functionality with call to FCOT. Upon detecting misbehavior, the cheated party demands all parties to open all of their FCOTs such that every party can retrace the entire computation.[6] We prove this formally in Appendix A.5.

## 3 Technical Overview

***Intuition.*** Most ID-MPC protocols are based on identifiable secret sharings [35, 36, 48]. This approach has a long history going back to [44] and as such it lead to many theoretical and practical results [44, 35, 36, 3, 27, 4, 48, 15].

We propose an alternative approach. The basic idea is that honest parties publicly announce conflicts with parties that are caught cheating. Intuitively, the conflicts are edges on a graph with one vertex per party. We call this graph the Conflict Graph (CG). The main conceptual advantage of the CG is that the abort-condition directly translates into publicly verifiable properties of the CG.

***Outline.*** Let us outline the rest of this section. First we give a formal definition of the CG and its ideal functionality followed by a description of the "proper" usage of said functionality. Then we state the relevant graph-theoretical properties and link the to the abort-condition of protocols using the CG is the described way. Next, we state a lemma about the graph's properties which—by the previous connection—translates into restrictions on the adversary ability to abort setups. Finally, we can use the lemma in our constructions: in a first step, we extend a commitment from $n-1$ parties to $n$ parties. In a second step, we extend FCOT from $n-1$ parties to $n$ parties using the global commitment constructed in the first step. Lastly, we show that FCOT, SFE, and Correlated-Randomness for $n$ parties imply each other.

Additionally, we briefly discuss the repeated application of our expansion protocol and its limitations.

***Formal Conflict Graph.*** Let us start with the formal definition of the CG.

**Definition 3 (Conflict Graph).** *The* Conflict Graph *(CG) is defined as the output of the following functionality.*

[6] Care has to be take to ensure that no sensitive information leaks to the adversary.

*Functionality $\mathcal{F}_{\mathsf{CG}}^n$*

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

- *Upon first activation, initiate the set of conflict edges $E := \emptyset$.*
- *When receiving a message $(\texttt{conflict}, \mathsf{P}_i \in P)$ from $\mathsf{P}_j$, for $i \neq j$ append the new conflict edge $\{\mathsf{P}_i, \mathsf{P}_j\}$ to the set of conflict 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 $\mathsf{P}_j$, for each $\mathsf{P}_i \in P'$ s.t. $i \neq j$ append the new conflict edge $\{\mathsf{P}_i, \mathsf{P}_j\}$ to $E$ and send $(\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i)$ to the adversary. (This instruction is stated merely for notational ease of conflicts with multiple parties.)*
- *When receiving a message $(\texttt{query})$ from $\mathsf{P}_j$, output the* Conflict Graph *$G := (P, E)$ to $\mathsf{P}_j$.*
- *When receiving $(\texttt{abort}, C')$ from $\mathcal{S}$ with $\emptyset \neq C' \subseteq C$, then output $(\texttt{abort}, C')$ to all parties and terminate.*

*The object of the Conflict Graph $G := (P, E)$ itself is an undirected irreflexive graph on all parties returned by the $\mathcal{F}_{\mathsf{CG}}^n$ functionality.*

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

**Lemma 2.** *Let $n$ be the number of parties of which at most $0 \leq t < n$ are malicious. There is a protocol $\pi_{\mathsf{CG}}$ in the $\{\mathcal{F}_{\mathsf{BC}}^n\}$-hybrid model that statistically UC-realizes $\mathcal{F}_{\mathsf{CG}}^n$:*

$$\mathcal{F}_{\mathsf{BC}}^n \overset{\text{perf}}{\rightsquigarrow} \mathcal{F}_{\mathsf{CG}}^n \tag{5}$$

We prove this is Appendix A.1. To link the abort-condition of a protocol to graph properties we want the protocol to ensure that honest parties are never in conflict. This is in particular fulfilled by the following definition.

**Definition 4 (Abort-respecting protocols).** *A protocol with parties $P$ and corrupted parties $C \subseteq P$ in some $F$-hybrid model s.t. $\mathcal{F}_{\mathsf{CG}}^n \in F$ is* abort-respecting,[7] *iff its description contains the following rules:*

*Rule* 1 *At the onset of each round, each party queries $\mathcal{F}_{\mathsf{CG}}^n$ with $(\texttt{query})$.*

*Rule* 2 *Whenever a setup functionality with parties $P'$ is aborted with $(\texttt{abort}, C')$ s.t. $\emptyset \neq C' \subseteq C \cap P'$, each honest party in $P' \setminus C'$ inputs $(\texttt{conflict}, C')$ into $\mathcal{F}_{\mathsf{CG}}^n$. Set $D' := C'$. At the onset of the next round when obtaining $G := (P, E)$ honest parties extract the set $L$ of "loyal" parties in $P'$ that are not in conflict with all parties in $D'$. The honest parties declare a conflicts with loyal parties by sending $(\texttt{conflict}, L)$ to $\mathcal{F}_{\mathsf{CG}}^n$. If the subgraph on $P'$ is not (yet) biseparated (as defined in Definition 6), then add $D' \leftarrow D' \cup L$ and repeat the same proceed in the next round.*
*Honest parties defer processing all other inputs until biseparation of the subgraph is reached.*

*Rule* 3 *Whenever an honest party is about to abort the protocol with $(\texttt{abort}, C')$ s.t. $\emptyset \neq C' \subseteq C$ and the overall CG is not yet biseparated, it inputs $(\texttt{conflict}, C')$ into $\mathcal{F}_{\mathsf{CG}}^n$. Furthermore, the honest parties proceed analogously to Rule 2 by gradually declaring conflicts with remaining loyalists until the overall CG becomes biseparated.*

*Rule* 4 *When the overall CG becomes biseparated, each party $\mathsf{P}$ determines its connected component $P' \ni \mathsf{P}$ in the complement CG $\overline{G}$, e.g. via breadth-first-search, and aborts with $(\texttt{abort}, P \setminus P')$. Additionally, if the CG contains some party $\mathsf{P}'$ with strictly more than $t$ conflicts, then the honest parties sends $(\texttt{conflict}, \mathsf{P}')$ to $\mathcal{F}_{\mathsf{CG}}^n$. The honest parties then perform the same procedure as in Rule 2 to gradually declare conflicts with remaining loyalists until the overall CG becomes biseparated.*

*Rule* 5 *Honest parties only declare 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 5, i.e., there are no honest-honest conflicts. In particular, generic conflicts fulfill this requirement honest parties only declare a conflicts

[7] [36] also use the term *abort-respecting*, we generalize the notion to the context of Conflict Graphs.

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 by assumption all identified parties upon abort 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 $\mathsf{P}$ gets more than $t$ conflicts for the first time in some fixed round $\rho$. Consequently, all (strictly more than $t$) parties in conflict with $\mathsf{P}$ must be malicious because before round $\rho$ conflicts were only issued due to Rule 2 and Rule 3. This, however, contradicts the assumption that at most $t$ parties are malicious, hence $\mathsf{P}$ 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 $D'$.

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 Appendices A.3 and A.4.
We emphasize that Definition 4 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 $\mathsf{P}$ instead broadcasts $\perp$ or nothing, then $\mathsf{P}$ must be malicious and it is safe to declare a (specific) conflict with $\mathsf{P}$ because $\mathsf{P}$ violated the protocol specification in an obvious manner. Looking ahead to the next lemma, we note that a protocol can canonical be made abort-respecting by amending the protocol with instructions (Rule 1, Rule 2, Rule 3 and Rule 4) corresponding to generic conflicts but none for specific conflicts.

**Lemma 3 (CG augmentation).** *Let $n$ be the number of parties $P$ of which at most $0 \leq t < n$ are malicious. Let $\pi^F$ be a protocol that securely UC-realizes a functionality $\mathcal{F}^n$ in some model $F \not\ni \mathcal{F}_{\mathsf{CG}}^n$. Denote by $\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n}$ the $F \cup \{\mathcal{F}_{\mathsf{CG}}^n\}$-hybrid protocol that is identical to $\pi^F$ with the addition that it uses $\mathcal{F}_{\mathsf{CG}}^n$ in an* abort-respecting *way according to Definition 4. Then $\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n}$ also securely UC-realizes $\mathcal{F}^n$, i.e., $\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n} \geq \mathcal{F}^n$.*

*Proof.* Here, the intuition is that the environment can infer the CG from its own behavior without querying $\mathcal{F}_{\mathsf{CG}}^n$. Furthermore, the additional behavior described in Definition 4 is deterministic and can be inferred from a transcript of the base protocol $\pi^F$ by adding the appropriate calls to $\mathcal{F}_{\mathsf{CG}}^n$ and potentially aborting, i.e., ending the protocol with abort-messages, if Rule 4 in Definition 4 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 $\pi^F$ securely UC-realizes $\mathcal{F}^n$.[8] That is $\mathcal{S}$ is the simulator for protocol $\pi^F$.
Let $\widetilde{\mathcal{Z}}$ denote an environment for protocol $\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n}$. Let $\mathcal{Z}_*[\widetilde{\mathcal{Z}}]$ be the environment for protocol $\pi^F$ that internally emulates $\widetilde{\mathcal{Z}}$. As the inner environment $\widetilde{\mathcal{Z}}$ expects an interface to $\mathcal{F}_{\mathsf{CG}}^n$ the outer environment $\mathcal{Z}$ simulates a $\mathcal{F}_{\mathsf{CG}}^n$ setup for it. $\mathcal{Z}$ outputs whatever $\mathcal{Z}_*[\widetilde{\mathcal{Z}}]$ outputs with the following modification: $\mathcal{Z}$ also simulates the additional behavior of honest parties as described in Definition 4 using the internally simulated $\mathcal{F}_{\mathsf{CG}}^n$. As such, the output transcript of $\mathcal{Z}_*[\widetilde{\mathcal{Z}}]$ when running with $\pi^F$ is looks exactly like the output of $\widetilde{\mathcal{Z}}$ when playing with $\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n}$, i.e.,

$$\mathsf{REAL}_{\pi^F,\mathcal{A}_\mathcal{D}}(\mathcal{Z}_*[\widetilde{\mathcal{Z}}]) \equiv \mathsf{REAL}_{\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n},\mathcal{A}_\mathcal{D}}(\widetilde{\mathcal{Z}}) \ . \tag{6}$$

Furthermore, let $\widetilde{\mathcal{S}}$ be the simulator that internally emulates $\mathcal{S}$ and also simulates an $\mathcal{F}_{\mathsf{CG}}^n$ setup. Analogously to $\widetilde{\mathcal{Z}}$, the outer simulator $\widetilde{\mathcal{S}}$ outputs whatever the inner simulator $\mathcal{S}$ outputs with the modification that $\widetilde{\mathcal{S}}$ also simulates the additional behavior of honest parties as described in Definition 4 using the internally simulated $\mathcal{F}_{\mathsf{CG}}^n$, hence

$$\mathsf{IDEAL}_{\mathcal{F}^n,\mathcal{S}}(\mathcal{Z}_*[\widetilde{\mathcal{Z}}]) \equiv \mathsf{IDEAL}_{\mathcal{F}^n,\widetilde{\mathcal{S}}}(\widetilde{\mathcal{Z}}) \ . \tag{7}$$

[8] As in Notation 5, $\mathcal{A}_\mathcal{D}$ is the canonical dummy adversary.

Now, suppose for contradiction that $\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n}$ does not realize $\mathcal{F}^n$, then

$$\begin{aligned} &\forall \widetilde{\mathcal{S}}' \, \exists \widetilde{\mathcal{Z}}' : \mathsf{REAL}_{\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n},\mathcal{A}_{\mathcal{D}}}(\widetilde{\mathcal{Z}}') \not\approx \mathsf{IDEAL}_{\mathcal{F}^n,\widetilde{\mathcal{S}}'}(\widetilde{\mathcal{Z}}') \\ \overset{\text{Eqs. (6)and (7)}}{\Longrightarrow} &\exists \widetilde{\mathcal{Z}} : \mathsf{REAL}_{\widetilde{\pi}^{F,\mathcal{F}_{\mathsf{CG}}^n},\mathcal{A}_{\mathcal{D}}}(\widetilde{\mathcal{Z}}) \not\approx \mathsf{IDEAL}_{\mathcal{F}^n,\widetilde{\mathcal{S}}}(\widetilde{\mathcal{Z}}) \\ \Longrightarrow\ &\mathsf{REAL}_{\pi^F,\mathcal{A}_{\mathcal{D}}}(\mathcal{Z}_*[\widetilde{\mathcal{Z}}]) \not\approx \mathsf{IDEAL}_{\mathcal{F}^n,\mathcal{S}}(\mathcal{Z}_*[\widetilde{\mathcal{Z}}]) \ . \end{aligned} \tag{8}$$

This contradicts the assumption that $\pi^F$ securely realizes $\mathcal{F}^n$, i.e.,

$$\begin{aligned} &\forall \mathcal{Z}' : \mathsf{REAL}_{\pi^F,\mathcal{A}_{\mathcal{D}}}(\mathcal{Z}') \approx \mathsf{IDEAL}_{\mathcal{F}^n,\mathcal{S}}(\mathcal{Z}') \\ \Longrightarrow\ &\mathsf{REAL}_{\pi^F,\mathcal{A}_{\mathcal{D}}}(\mathcal{Z}_*[\widetilde{\mathcal{Z}}]) \approx \mathsf{IDEAL}_{\mathcal{F}^n,\mathcal{S}}(\mathcal{Z}_*[\widetilde{\mathcal{Z}}]) \ . \end{aligned} \tag{9}$$

□

***Graph properties.*** Supposing that honest parties are not in conflict,[9] we call any subset of parties which could have caused the Conflict Graph $G$ an *explanation* of $G$. Essentially, an explanation of a Conflict Graph $G$ is a *Vertex Cover* (VC) of $G$. We differentiate between *internal* and *external* explanations. External explanations allow even non-participants to identify a set of malicious parties, given only the Conflict Graph and the maximum number $t$ of corrupted parties. This abort-condition corresponds to the following graph property.

**Definition 5 ($t$-settledness).** *Let $n \geq 1$ be the number of parties of which at most $0 \leq t < n$ are malicious. Let $G = (P, E)$ be the Conflict Graph of some $F$-hybrid protocol $\pi$ s.t. $\mathcal{F}_{\mathsf{CG}}^n \in F$ where honest parties are never in conflict. Let $M(G, t)$ be the set of all VCs (explanations) of $G$ with size $t$ or less, and let $X(G, t)$ be the intersection of all of these VCs,[10] that is, the set of parties which are present in all VCs of size at most $t$. We call $X(G, t)$ the* settled set *of $G$. We call $G$ $t$-*settled*, iff $X(G, t) \neq \emptyset$.*

This definition precisely captures the equivalence between $t$-settledness of a Conflict Graph and its *external explanation*—each (Minimum) Vertex Cover of size at most $t$ is a valid explanation of corrupted parties that *could* have caused this graph (pattern of conflicts). If some party $\mathsf{P}$ is contained in each possible explanation, then surely $\mathsf{P}$ cannot be honest. Even non-participants can be convinced, using only the CG, that party $\mathsf{P}$ is malicious.

*Remark 1.* In general, it might be hard to compute the settled set $X(G, t)$ since the naive way of computing it requires intersecting $\binom{n}{t}$ many VCs, or alternatively finding Minimum VCs. Indeed, in Appendix B we show that finding the settled set of a general graph $G$ is as hard as finding a Minimum VC of $G$.

To avoid confusion, we stress that our construction does not require finding the settled set of the Conflict Graph.

Now we have seen that $t$-settledness of the Conflict Graph allows for external observers to identify malicious parties, albeit at the cost of computing many Vertex Covers. For Identifiable Abort, it is usually not necessary for participants to convince external parties, but only participants must be able to identify malicious parties. We therefore introduce a different graph property which reflects valid explanations for participants only:

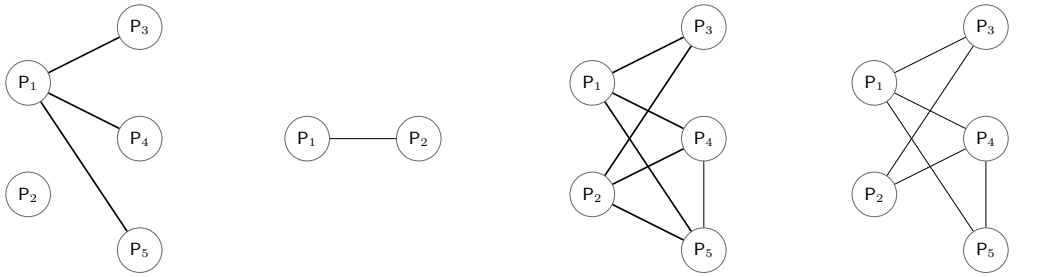
**Definition 6 (Biseparation).** *A Conflict Graph $G = (P, E)$ is called* biseparated*, iff there exists a subset $E' \subseteq E$ that forms a complete bipartite graph (biclique) on $P$.*

Here the intuition is that there exist two partitions $P_1 \cup P_2 = P$ s.t. each party in $P_1$ is in conflict with each party in $P_2$. Because honest parties are never in conflict, all honest parties must be contained in the same partition. As such all parties in one partition can identify the parties in the other partition as malicious; though to outsiders it may not be clear which partition is contains the honest parties.

*Remark 2.* A Conflict Graph $G = (P, E)$ is biseparated, iff its Complement Graph $\overline{G} = (P, \overline{E})$ is disconnected.

[9] This property must be ensured by the protocol using the Conflict Graph. In particular, abort-respecting protocols fulfill this property.

[10] The intersection of all VCs is exactly the intersection of all Minimum Vertex Covers.




**(a)** 1-settled example **(b)** 1-settled counterexample **(c)** Biseparated example **(d)** Biseparated counterexample

**Fig. 2:** Several (counter-)examples for the introduced conflict graph conditions. Thick lines are relevant for the respective property.

This follows directly from the fact that the complement graph of a biclique contains no path from one partition to the other.

To give an intuition of the graph properties, a few example graphs are shown in Fig. 2. The graph in Fig. 2a has only one vertex cover of size 1, namely $\{\mathsf{P}_1\}$, hence $X(G,1) = \{\mathsf{P}_1\}$. Therefore $\mathsf{P}_1$ is in all external explanations, making the graph 1-settled. Note that this graph is *not* biseparated. The graph from Fig. 2b on the other hand is not 1-settled; both $\{\mathsf{P}_1\}$ and $\{\mathsf{P}_2\}$ are valid Vertex Covers. Biseparation trivially holds. Another example for biseparation is the Conflict Graph from Fig. 2c. This graph can be split up into two partitions $P_1 \coloneqq \{\mathsf{P}_1, \mathsf{P}_2\}$ and $P_2 \coloneqq \{\mathsf{P}_3, \mathsf{P}_4, \mathsf{P}_5\}$. This implies that all members of $P_2$ are convinced that $\mathsf{P}_1$ and $\mathsf{P}_2$ are corrupted. However, if we remove the conflict between $\mathsf{P}_2$ and $\mathsf{P}_5$ (see Fig. 2d), the graph is no longer biseparated, thus $\mathsf{P}_5$ would not identify $\mathsf{P}_2$ as malicious.
In particular, Fig. 2a nicely demonstrates that—in general—$t$-settledness (external explanation) does *not* imply[11] biseparation (internal explanation).

Now that we have a good intuition of the presented graph properties, we can formally show that the notion of biseparation corresponds to Identifiable Abort.

**Lemma 4 (Abort $\iff$ biseparation).** *Let $\pi$ be an abort-respecting protocol that securely UC-realizes a functionality $\mathcal{F}^n$ with Identifiable Abort in an $F \cup \{\mathcal{F}_{\mathsf{CG}}^n\}$-hybrid model.*

1. *If the CG of $\pi$ becomes biseparated the (honest) parties abort by identifying malicious parties.*
2. *Upon abort, the CG $G$ of $\pi$ must be biseparated.*

*Proof.* The first statement follows directly from Rule 4 in Definition 4. 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 complement CG. 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 3 in Definition 4. □

Now, we have formally linked the abort-condition to the graph-theoretical properties of the CG. We go ahead and focus on the special case of $t \leq n-3$ where we are given a guarantee that *at least* three parties are honest. Effectively, the line of argument is as follows: suppose the adversary aborts setups is a certain way, hence the resp. subgraphs become biseparated. Then the the graph-theoretical lemma states that the overall Conflict Graph becomes biseparated, hence the protocol can abort. This way we can limit the adversary's power of aborting setups.

In the following, we use the notation $\mathcal{F}_k^n$ for a setup on parties $P \setminus \{\mathsf{P}_k\}$ to prove a technical lemma.

**Lemma 5.** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious. Let $\pi$ be an abort-respecting protocol that securely UC-realizes a functionality $\mathcal{F}^n$ with Identifiable Abort in an $\{\mathcal{F}^{n-1}, \mathcal{F}_{\mathsf{CG}}^n\}$-hybrid model where $\mathcal{F}^{n-1}$ stands for any setup of size $n-1$. Let $\mathsf{P}_i, \mathsf{P}_j \in P$ be two different parties whose respective setups $\mathcal{F}_i^{n-1}$ and $\mathcal{F}_j^{n-1}$ are aborted. Moreover, for any aborted $\mathcal{F}_k^{n-1}$ let $D_k$—the* disruptor set*—be the largest partition of parties s.t. $P \setminus \{\mathsf{P}_k\}$ is biseparated with $D_k$ and $P \setminus (\{\mathsf{P}_k\} \cup D_k)$.*

*If $\mathsf{P}_j \in D_i$ and the overall Conflict Graph is not biseparated, then $D_j \subset D_i$.*

[11] The implication in this sense means that the given $t$-settled graph *itself* is not biseparated. However a $t$-settled graph can be (possibly inefficiently) transformed into a biseparated graph. We elaborate on this in Appendix B.

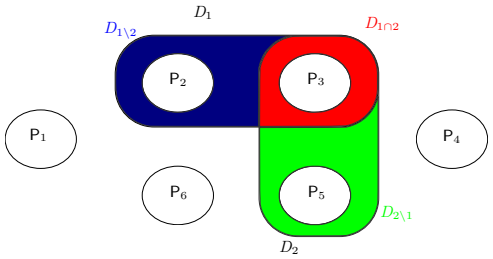



**Fig. 3:** Depiction of the disruptor sets for six parties.

*Proof.* Recall that after any abort of a setup $\mathcal{F}_k^{n-1}$ that omits $\mathsf{P}_k$ the subgraph on parties $P \setminus \{\mathsf{P}_k\}$ is biseparated between $P \setminus \{\mathsf{P}_k \cup D_k\}$ and $D_k$, where $D_k$ contains all parties that actively were caught cheating and those that did not declare a conflict in Rule 2 in Definition 4. Let's assume for the sake of contradiction that $D_j$ is *not* a subset of $D_i$, i.e., $D_i \not\subseteq D_j \iff D_j \setminus D_i \neq \emptyset$ because $D_i \neq D_j$. For short notation we define

$$D_{i\cap j} \coloneqq D_i \cap D_j \ , \quad D_{i\setminus j} \coloneqq D_i \setminus D_j \ , \quad D_{j\setminus i} \coloneqq D_j \setminus D_i \ , \quad D_{i\cup j} \coloneqq D_j \cup D_i \ . \tag{10}$$

We also write $A \nsim B$ if all parties in the set $A$ are in conflict with all parties in set $B$.

An example disruptor set distribution this is shown in Fig. 3. There $\mathsf{P}_2$ and $\mathsf{P}_3$ abort $\mathcal{F}_{\mathsf{P}_1}^5$ and $\mathsf{P}_3$ and $\mathsf{P}_5$ abort $\mathcal{F}_{\mathsf{P}_2}^5$. The cut $D_{1\cap 2}$ between the two sets is depicted in red and contains the party that aborts both setups. The two remaining sets are given in blue for $D_{1\setminus 2}$ of the party that aborts $\mathcal{F}_{\mathsf{P}_1}^5$ but acts honest in $\mathcal{F}_{\mathsf{P}_2}^5$ and in green for $D_{2\setminus 1}$ of the party that aborts $\mathcal{F}_{\mathsf{P}_2}^5$ but acts honestly in $\mathcal{F}_{\mathsf{P}_1}^5$.

Moving back to the general case, consider $\mathsf{P}_i \notin D_j$. Now, we make the observation that

$$\{\mathsf{P}_j\} \cup D_{i\cap j} \overset{\mathsf{P}_j \in D_{i\setminus j}}{\subseteq} D_i \nsim P \setminus (\{\mathsf{P}_i\} \cup D_i) \overset{\mathsf{P}_i \notin D_j}{\supseteq} D_{j\setminus i} \tag{11}$$

due to the abort of $\mathcal{F}_i^{n-1}$. Similarly, we see that

$$D_{j\setminus i} \subseteq D_j \nsim P \setminus (\{\mathsf{P}_j\} \cup \underbrace{D_j}_{D_{i\cap j} \cup D_{j\setminus i}}) = P \setminus \left(\{\mathsf{P}_j\} \cup D_{i\cap j} \cup D_{j\setminus i}\right) \tag{12}$$

due to the abort of $\mathcal{F}_j^{n-1}$. Combining the two sets of conflicts we obtain

$$D_{j\setminus i} \nsim P \setminus \left(\{\mathsf{P}_j\} \cup D_{i\cap j} \cup D_{j\setminus i}\right) \cup \{\mathsf{P}_j\} \cup D_{i\cap j} = P \setminus D_{i\setminus j} \ . \tag{13}$$

Therefore the overall CG is biseparated on the partitions $D_{j\setminus i}$ and $P \setminus D_{j\setminus i} \neq \emptyset$ by our counter assumption. This stands in contradiction to the lemma's requirement that the Conflict Graph must not be biseparated.

Now, consider the remaining case $\mathsf{P}_i \in D_j$. Then

$$D_i \nsim P \setminus (\{\mathsf{P}_i\} \cup D_i) \overset{\mathsf{P}_i \in D_j}{\supseteq} P \setminus D_{i\cup j} \tag{14}$$

due to the abort of $\mathcal{F}_i^{n-1}$. Similarly, we see that

$$D_j \nsim P \setminus (\{\mathsf{P}_j\} \cup D_j) \overset{\mathsf{P}_j \in D_i}{\supseteq} P \setminus D_{i\cup j} \tag{15}$$

due to the abort of $\mathcal{F}_j^{n-1}$. Combining the two sets of conflicts we obtain

$$P \setminus D_{i\cup j} \nsim D_{i\cup j} \tag{16}$$

which means the CG is biseparated on the partitions $D_{i\cup j}$ and $P \setminus D_{i\cup j}$. Moreover, $P \setminus D_{i\cup j} \neq \emptyset$ because it contains all honest parties. As in the previous case, this contradicts the lemma's requirement. □

Now we have all the tools necessary to effectively limit the amount of setups that can be aborted if at least three parties are honest.

**Lemma 6.** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious. Let $\pi$ be an abort-respecting protocol that securely UC-realizes a functionality $\mathcal{F}^n$ with Identifiable Abort in an $\{\mathcal{F}^{n-1}, \mathcal{F}_{\mathsf{CG}}^n\}$-hybrid model where $\mathcal{F}^{n-1}$ stands for any setup of size $n-1$.*

*If $t$ or more setups of size $(n-1)$ are aborted, then the Conflict Graph is biseparated.*

*Proof.* Since the proof itself is quite technical, we want to give an intuition first. Each abort of a setup requires revealing the identity of at least one malicious party. Thus, intuitively, each abort "costs" the adversary at least one malicious party. Hence, at most $t$ setups can be aborted.
The complement Conflict Graph looks like a generalized star graph with the $(n-t)$-clique of honest parties as the "core" and one path originating from each honest party as the "arms". Hence, at least one party (terminal party of the path), has only one neighbor in the complement CG. In other words, these terminals have $n-2 > n-3 \geq t$ conflicts. Therefore, these terminals would be identified as malicious.

We denote by $P \coloneqq \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ the set of parties. For each party $\mathsf{P}_i$ there is a setup $\mathcal{F}_i^{n-1}$ of size $n-1$ that excludes $\mathsf{P}_i$. For any aborted $\mathcal{F}_i^{n-1}$ with $(\texttt{abort}, C')$ let $D_i \supseteq C'$ be the largest partition of parties s.t. $P \setminus \{\mathsf{P}_i\}$ is biseparated with $D_i$ and $P \setminus (\{\mathsf{P}_i\} \cup D_i)$, we call the $D_i$ *disruptor sets*. We write $D_i = \emptyset$ if $\mathcal{F}_i^{n-1}$ terminated successfully. In this case, we call $\mathsf{P}_i$ a *terminal* party. Let $I_i \coloneqq \{j \mid \mathsf{P}_j \in D_i\}$ denote the set of indices corresponding to parties in $D_i$, and let $N_j \coloneqq |I_j|$ for short notation. Recall from Lemma 5 that

$$\forall \mathsf{P}_j \in D_i : D_j \subset D_i \ , \quad \text{or equivalently} \quad \forall j \in I_i : I_j \subset I_i \ . \tag{17}$$

Our goal is to show that for each honest party $\mathsf{P}_i$ there exists a (distinct) terminal party $\mathsf{P}_{\tau(i)}$ s.t. $\tau(i) \in I_i \cup \{i\}$ whose setup $\mathcal{F}_{\tau(i)}^{n-1}$ is not aborted, i.e., $N_{\tau(i)} = 0$.
Suppose for contradiction that $\forall j \in I_i : N_j \geq 1$. This means that the map $\rho : I_i \to I_i : j \mapsto \min(I_j)$ is fix-point-free and

$$\forall j \in I_i : \rho(j) = \min(I_j) \in I_j \tag{18}$$

$$\overset{\text{Eq. (17)}}{\implies} \forall j \in I_i : I_{\rho(j)} \subset I_j \tag{19}$$

$$\implies \forall j \in I_i : N_{\rho(j)} < N_j \tag{20}$$

holds. For each $k \geq 0$ let $\rho^k$ denote the $k$-wise composition of $\rho$, i.e., evaluating the map $k$ times. Starting from any disruptor's index $\jmath \in I_i$ we consider the list $(\rho^k(\jmath))_{k \geq 0}$. From Eq. (20) it follows that

$$\forall k \geq 0 : N_{\rho^{k+1}(\jmath)} < N_{\rho^k(\jmath)} \tag{21}$$

$$\implies \forall k \geq 0, \ell > 1 : N_{\rho^{\ell+k}(\jmath)} < N_{\rho^k(\jmath)} \ . \tag{22}$$

Because the domain $I_i$ is finite and $\rho$ is fix-point-free, this list must run into a cycle of length $\hat{\ell} \geq 2$. That is, there exists some index $\hat{\jmath} = \rho^{\hat{k}}(\jmath)$, resp. $\hat{k} \geq 0$, s.t. $\rho^{\hat{\ell}}(\hat{\jmath}) = \hat{\jmath}$. Consequently,

$$\rho^{\hat{\ell}+\hat{k}}(\jmath) = \rho^{\hat{\ell}}(\hat{\jmath}) = \hat{\jmath} = \rho^{\hat{k}}(\jmath) \implies N_{\rho^{\hat{\ell}+\hat{k}}(\jmath)} = N_{\rho^{\hat{k}}(\jmath)} \tag{23}$$

which contradicts Eq. (22). Thus, for each honest party $\mathsf{P}_i$ there exists a (distinct) *terminal* party $\mathsf{P}_{\tau(i)}$ s.t. $D_{\tau(i)} = \emptyset$.

Next we show that the terminal parties are actually distinct, i.e., each honest party has its own terminal party (or parties). Formally, we show that for each two (different) honest parties $\mathsf{P}_{i_1}, \mathsf{P}_{i_2}$ their disruptor sets $D_{i_1}, D_{i_2}$ are disjoint. For contradiction, suppose that $I \coloneqq D_{i_1} \cap D_{i_2} \neq \emptyset$. From abort of $\mathcal{F}_{i_1}^{n-1}$ we know that $I$ is in conflict with $P \setminus (\{\mathsf{P}_{i_1}\} \cup D_{i_1})$. From abort of $\mathcal{F}_{i_2}^{n-1}$ we know that $I$ is in conflict with $P \setminus (\{\mathsf{P}_{i_2}\} \cup D_{i_2})$. Overall, $I$ is in conflict with

$$\begin{aligned} & P \setminus ((\{\mathsf{P}_{i_1}\} \cup D_{i_1}) \cap (\{\mathsf{P}_{i_2}\} \cup D_{i_2})) \\ =& P \setminus (\underbrace{\{\mathsf{P}_{i_1}\} \cap \{\mathsf{P}_{i_2}\}}_{\emptyset} \cup \underbrace{D_{i_1} \cap \{\mathsf{P}_{i_2}\}}_{\emptyset} \cup \underbrace{\{\mathsf{P}_{i_1}\} \cap D_{i_2}}_{\emptyset} \cup \underbrace{D_{i_1} \cap D_{i_2}}_{I}) \\ =& P \setminus I \ , \end{aligned} \tag{24}$$

thus the overall CG would be biseparated with $I$ and $P \setminus I$. Therefore the terminals of each honest party are actually distinct.

Because there exist at least $n-t$ honest parties and their resp. terminal parties are distinct, there also exist at least $n-t$ setups that are not aborted. Conversely, up to $a \leq t$ setups can be aborted without causing a biseparation of the overall Conflict Graph. The lemma's assumption $a \geq t$ then implies that $a = t$ is the only possibility left. This means, that for each of the at least $n-t$ honest parties $\mathsf{P}_i$ there exists exactly one terminal party, either $\mathsf{P}_i$ itself, or some malicious party $\mathsf{P}_{\tau(i)} \in D_i$. Denote by $0 \leq t' \leq t$ the actual number of corrupted parties. If $t' = 0$, obviously no setup can be aborted. If $t' \geq 1$, then for at least one honest party $\mathsf{P}_i$ the corresponding terminal

must be a malicious party $\mathsf{P}_{\tau(i)} \in D_i$. For the honest party $\mathsf{P}_i$ let $W \coloneqq \{\mathsf{W}_1 = \mathsf{P}_i, ..., \mathsf{W}_w = \mathsf{P}_{\tau(i)}\}$ the "arm" path in the complement CG originating from $\mathsf{P}_i$ to its terminal $\mathsf{P}_{\tau(i)}$ of length $w \geq 2$. Since $\mathsf{P}_{\tau(i)}$ is only not in conflict with itself and $\mathsf{W}_{w-1}$, it must have $n-2 > n-3 \geq t$ conflicts which implies the biseparation of the CG by Rule 4 of Definition 4 in contradiction to the lemma's requirement. This contradiction concludes the proof. □

***Constructions.*** The next two results constitute the heart of our expansion protocol for $n$-party MPC from $(n-1)$-party setups. We prove these formally in Appendices A.3 and A.4 respectively. For a visual representation of our construction see Fig. 4.

**Theorem 1.** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious. There exists a protocol $\pi_{\mathsf{COM}}$ in the $\{\mathcal{F}_{\mathsf{COM}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^{n}\}$-hybrid model that statistically UC-realizes $\mathcal{F}_{\mathsf{COM}}^{n}$:*

$$\{\mathcal{F}_{\mathsf{COM}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^{n}\} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}_{\mathsf{COM}}^{n} \tag{25}$$

Without going into too much detail we give an outline of the protocol here.
To commit, the sender inputs a secret-sharing of its message into all commitment setups. The receivers broadcast probing indices whereupon the sender opens the requested shares. This ensures that all setups actually contain sharing that encode the same message.
If all probed shares are consistent, then the receivers acknowledge the commitment.
If any inconsistencies occur, i.e., receivers disagree on the message opened in a specific setup, then the parties declare conflicts with each other, thus biseparating the sub-CG on the parties that participated in the setup. Such a setup and its opened message is considered invalid. If the messages in all valid setups are equal, then each receiver outputs that message, otherwise they identify the sender as malicious and abort. Because at least one honest party participates in each setup the agreed (valid) message must be equal to the opened value of that setup, thus malicious receivers cannot frame the sender.

To open, the sender simply opens all setups s.t. all receivers learn the output. Then the receivers broadcast the shares that they received to check for inconsistencies. If no inconsistencies occur, all receivers output the same reconstructed message. Otherwise, if the sharings are consistent but decode to $\perp$, then the receivers identify the sender as malicious.

**Theorem 2.** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious. There exists a protocol $\pi_{\mathsf{FCOT}}$ in the $\{\mathcal{F}_{\mathsf{FCOT}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^{n}\}$-hybrid model that statistically UC-realizes $\mathcal{F}_{\mathsf{FCOT}}^{n}$:*

$$\{\mathcal{F}_{\mathsf{FCOT}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^{n}\} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}_{\mathsf{FCOT}}^{n} \tag{26}$$

The high-level description of the protocol is as follows: the sender samples masks for its messages and creates sharings of its masks and masked messages. The sender globally commits to all shares individually. The receiver creates a sharing for its choice bit and globally commits to the shares. Then the sender and the receiver input their sharings into all $\mathcal{F}_{\mathsf{FCOT}}^{n-1}$ setups which perform an OT on the encoded values (formally we use an SFE setup for this, which can be realized using $\mathcal{F}_{\mathsf{FCOT}}^{n-1}$). The crucial point here is that the sender and the receiver open some uniformly random shares to prove that their inputs for the setups are indeed the same as the globally committed shares. If at least one setups succeeds, the sender opens the resp. mask and the receiver learns its chosen message. Note that the receiver only learns one message because the sender only opens one mask. If all setups are aborted, then the overall Conflict Graph becomes biseparated due to Lemma 6, and hence the honest parties can abort.

To open, the sender resp. receiver simply open their global commitments and each party can reconstruct the resp. value from all shares.

**Lemma 7.** *For every number of parties $n$, the functionalities $\mathcal{F}_{\mathsf{SFE},f}^{n}$, $\mathcal{F}_{\mathsf{FCOT}}^{n}$ and $\mathcal{F}_{\mathsf{Corr},\mathcal{D}}^{n}$ are equally powerful.*

$$\mathcal{F}_{\mathsf{FCOT}}^{n} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}_{\mathsf{SFE},f}^{n} \overset{\text{perf}}{\rightsquigarrow} \mathcal{F}_{\mathsf{Corr},\mathcal{D}}^{n} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}_{\mathsf{FCOT}}^{n} \tag{27}$$

We prove this in Appendix A.5. The first construction is essentially the IPS-compiler with FCOT setups. The second construction follows directly from the fact that $\mathcal{F}_{\mathsf{Corr},\mathcal{D}}^{n}$ is a special case of $\mathcal{F}_{\mathsf{SFE},f}^{n}$. The third construction follows from the MPC-completeness of the Correlated-Randomness model in [36].

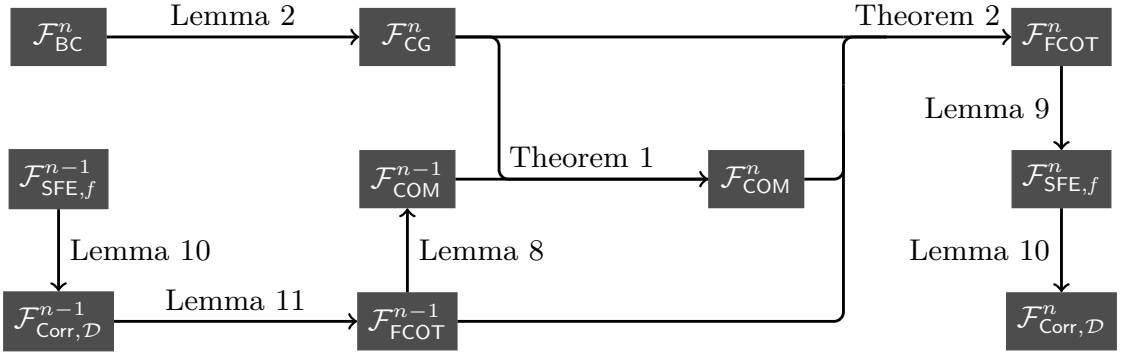



**Fig. 4:** An overview of the steps that result in Corollary 1.

**Corollary 1 (MPC expansion).** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious.*

$$\left\{ \begin{matrix} \mathcal{F}^{n-1}_{\mathsf{Corr},\mathcal{D}} \\ \mathcal{F}^{n-1}_{\mathsf{FCOT}} \\ \mathcal{F}^{n-1}_{\mathsf{SFE},f} \end{matrix} , \mathcal{F}^n_{\mathsf{BC}} \right\} \overset{\text{stat}}{\rightsquigarrow} \left\{ \begin{matrix} \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}} \\ \mathcal{F}^n_{\mathsf{FCOT}} \\ \mathcal{F}^n_{\mathsf{SFE},f} \end{matrix} \right\} \tag{28}$$

By extending the broadcast to a *multicast*,[12] where only a subset of parties receive the message, we can extend the result for any $t \leq n-3$ using recursion: if $t \leq n-4$, then $\mathcal{F}^{n-3}_{\mathsf{SFE},f}$ and $\mathcal{F}^{n-1}_{\mathsf{BC}}$ realize $\mathcal{F}^{n-2}_{\mathsf{SFE},f}$. Further application of this recursion implies that $\mathcal{F}^{t+2}_{\mathsf{SFE},f}$ and $\{\mathcal{F}^i_{\mathsf{BC}}\}^n_{i=t+3}$ realize $\mathcal{F}^n_{\mathsf{SFE},f}$.

However, since the overall runtime grows exponentially in the number of recursions we have to limit them by $\mathcal{O}(\ln\lambda/\ln\ln\lambda)$. For $n/2$ recursions we obtain an overall runtime of at most

$$(n)^c \cdot (n-1)^c \cdots (n/2)^c = (n!/(n/2-1)!)^c \tag{29}$$

where $c$ is some constant s.t. the runtime of a single expansion protocol is bounded by $n^c$. The above bound is polynomial in $\lambda$ if e.g. $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$.

**Corollary 2.** *For $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$ and up to $t \in \mathcal{O}(n)$ corruptions, it holds for the minimal complete cardinality that $k^* \leq t+2$ relative to $F = \{\mathcal{F}^i_{\mathsf{BC}} \mid t+3 \leq i \leq n\}$.*

**Corollary 3.** *For $n \in \text{poly}(\lambda)$ and up to $t$ corruptions s.t. $n - t \in \Theta(1)$, it holds for the minimal complete cardinality that $k^* \leq t+2$ relative to $F = \{\mathcal{F}^i_{\mathsf{BC}} \mid t+3 \leq i \leq n\}$.*

Finally, combining Corollary 13 of [36] with Corollary 2 yields

**Corollary 4.** *For $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$ and up to $t \in \mathcal{O}(n)$ corruptions,*

$$\left\{\pi^{\mathsf{OT}}, \mathcal{F}^{t+2}_{\mathsf{CRS}}, \mathcal{F}^{t+3}_{\mathsf{BC}}, ..., \mathcal{F}^n_{\mathsf{BC}}\right\} \overset{\text{comp}}{\rightsquigarrow} \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}} \tag{30}$$

*where $\pi^{\mathsf{OT}}$ is any adaptively secure OT protocol used in a black-box manner.*

## References


[1] M. Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *28th ACM STOC*. ACM Press, May 1996, pp. 99–108.

[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.


[12] The original formulation of [19] is actually a multicast.

[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] M. Ben-Or, S. Goldwasser, and A. Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 1–10.

[8] A. R. Block, H. K. Maji, and H. H. Nguyen. "Secure Computation Based on Leaky Correlations: High Resilience Setting". In: *CRYPTO 2017, Part II*. Ed. by J. Katz and H. Shacham. Vol. 10402. LNCS. Springer, Heidelberg, August 2017, pp. 3–32.

[9] A. R. Block et al. "Secure Computation Using Leaky Correlations (Asymptotically Optimal Constructions)". In: *TCC 2018, Part II*. Ed. by A. Beimel and S. Dziembowski. Vol. 11240. LNCS. Springer, Heidelberg, November 2018, pp. 36–65.

[10] D. Boneh. "The decision Diffie-Hellman problem". In: *Third Algorithmic Number Theory Symposium (ANTS)*. Vol. 1423. LNCS. Invited paper. Springer, Heidelberg, 1998.

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

[12] E. Boyle et al. "Efficient Pseudorandom Correlation Generators from Ring-LPN". In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, August 2020, pp. 387–416.

[13] 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.

[14] E. Boyle et al. "Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation". In: *ACM CCS 2019*. Ed. by L. Cavallaro et al. ACM Press, November 2019, pp. 291–308.

[15] N. Brandt. *Tight Setup Bounds for Identifiable Abort*. Cryptology ePrint Archive, Report 2021/684. https://ia.cr/2021/684. 2021.

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

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

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] R. K. Cunningham, B. Fuller, and S. Yakoubov. "Catching MPC Cheaters: Identification and Openability". In: *ICITS 17*. Ed. by J. Shikata. Vol. 10681. LNCS. Springer, Heidelberg, November 2017, pp. 110–134.

[28] 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.

[29] 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.

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

[31] 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.

[32] D. Gupta et al. “Secure Computation from Leaky Correlated Randomness”. In: *CRYPTO 2015, Part II*. Ed. by R. Gennaro and M. J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, August 2015, pp. 701–720.

[33] D. Hofheinz and J. Müller-Quade. “A Synchronous Model for Multi-Party Computation and the Incompleteness of Oblivious Transfer”. In: *IACR Cryptology ePrint Archive* 2004 (January 2004), p. 16.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] R. M. Karp. “Reducibility Among Combinatorial Problems”. In: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. Ed. by R. E. Miller and J. W. Thatcher. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103.

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

[41] 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.

[42] T. Moran, I. Orlov, and S. Richelson. “Topology-Hiding Computation”. In: *TCC 2015, Part I*. Ed. by Y. Dodis and J. B. Nielsen. Vol. 9014. LNCS. Springer, Heidelberg, March 2015, pp. 159–181.

[43] 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.

[44] 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.

[45] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *37th ACM STOC*. Ed. by H. N. Gabow and R. Fagin. ACM Press, May 2005, pp. 84–93.

[46] 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.

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

[48] M. Simkin, L. Siniscalchi, and S. Yakoubov. *On Sufficient Oracles for Secure Computation with Identifiable Abort*. Cryptology ePrint Archive, Report 2021/151. https://eprint.iacr.org/2021/151. 2021.

# Supplementary Material

## A Detailed Constructions

### A.1 Conflict Graph from Broadcast

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

**Lemma 2.** *Let $n$ be the number of parties of which at most $0 \leq t < n$ are malicious. There is a protocol $\pi_{\mathsf{CG}}$ in the $\{\mathcal{F}_{\mathsf{BC}}^n\}$-hybrid model that statistically UC-realizes $\mathcal{F}_{\mathsf{CG}}^n$:*

$$\mathcal{F}_{\mathsf{BC}}^n \overset{\text{perf}}{\rightsquigarrow} \mathcal{F}_{\mathsf{CG}}^n \tag{5}$$

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

Protocol $\mathcal{F}_{\mathsf{CG}}^n$

1. **At the onset** each party $\mathsf{P}_i \in P$ starts with a graph $G_i := (P, E_i)$ with $E_i := \emptyset$.
2. **On input** $(\texttt{conflict}, \mathsf{P}_i)$ from $\mathcal{Z}$ to $\mathsf{P}_j$, $\mathsf{P}_j$ inputs $(\texttt{conflict}, \mathsf{P}_i)$ to $\mathcal{F}_{\mathsf{BC}}^n$.
3. **On output** $(\texttt{output}, \mathsf{P}_j, (\texttt{conflict}, \mathsf{P}_i))$ from $\mathcal{F}_{\mathsf{BC}}^n$ 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 $\mathsf{P}_i$, $\mathsf{P}_i$ outputs it own $G_i$.
5. **On output** $(\texttt{abort}, C')$ from $\mathcal{F}_{\mathsf{BC}}^n$ each party outputs $(\texttt{abort}, C')$ and terminates.

A simulator for this protocol is straightforward:

Simulator for $\mathcal{F}_{\mathsf{CG}}^n$

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

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:

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

Hybrid $\mathcal{H}_2$ In this game there are two major changes from Hybrid $\mathcal{H}_1$: we introduce a dummy functionality $\widetilde{\mathcal{F}}_{\mathsf{CG}}^n$ that forwards all inputs to the simulator, and honest parties are replaced by dummy parties that forward their inputs to $\mathcal{F}_{\mathsf{CG}}^n$ and the corresponding simulator executes the code of the respective party. The simulator now also simulates the $\mathcal{F}_{\mathsf{BC}}^n$ 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.

Hybrid $\mathcal{H}_3$ Now the dummy functionality is replaced with the ideal functionality $\mathcal{F}_{\mathsf{CG}}^n$ and the simulator is correct simulator given above.
In particular, whenever a corrupted party $\mathsf{P}_j$ inputs $(\texttt{conflict}, \mathsf{P}_i)$ into the simulated $\mathcal{F}_{\mathsf{BC}}^n$ the simulator inputs $(\texttt{conflict}, \mathsf{P}_i)$ into $\mathcal{F}_{\mathsf{CG}}^n$ in the name of $\mathsf{P}_j$.
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 (`query`) as it involves no interaction. Honest parties merely forward the request and obtain the correct conflict graph $G$. And corrupted parties query their local copy. □

### A.2 Global Commitment from FCOT

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

**Lemma 8.** *There is a protocol $\pi_{\mathsf{COM}}$ that securely UC-realizes $\mathcal{F}_{\mathsf{COM}}^n$ in the $\{\mathcal{F}_{\mathsf{FCOT}}^n\}$-hybrid model:*

$$\mathcal{F}_{\mathsf{FCOT}}^n \overset{\text{perf}}{\rightsquigarrow} \mathcal{F}_{\mathsf{COM}}^n$$

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

Protocol $\pi_{\mathsf{COM}}$

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

Simulator for $\pi_{\mathsf{COM}}$

1. **On input** $(\texttt{messages}, m^0, m^1)$ from a corrupted sender $\mathsf{S}$ to $\mathcal{F}_{\mathsf{FCOT}}^n$, $\mathcal{S}$ reconstructs $m \coloneqq m^0 \oplus m^1$ and sends $(\texttt{commit}, m)$ to $\mathcal{F}_{\mathsf{COM}}^n$ in the name of $\mathsf{S}$.
2. **On output** $(\texttt{receipt commit})$ from $\mathcal{F}_{\mathsf{COM}}^n$, $\mathcal{S}$ samples two messages $m^0 \xleftarrow{\$} \{0,1\}^\lambda$ and $m^1 \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random and simulates $\mathcal{F}_{\mathsf{FCOT}}^n$ with sender input $(\texttt{messages}, m^0, m^1)$.
3. **On input** $(\texttt{choice}, c)$ for $c \in \{0,1\}$ by a corrupted receiver $\mathsf{R}_1$, $\mathcal{S}$ stores $c$ and reports $(\texttt{receipt transfer})$ to $\mathcal{Z}$ and simulates $\mathcal{F}_{\mathsf{FCOT}}^n$ 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 $\mathcal{F}_{\mathsf{COM}}^n$ in the name of the sender.
5. **On output** $(\texttt{open}, m)$ from $\mathcal{F}_{\mathsf{COM}}^n$, $\mathcal{S}$ equivocates the commitment by setting $m^{\overline{c}} \coloneqq m \oplus m^c$ for $\mathcal{F}_{\mathsf{FCOT}}^n$ and outputs $(\texttt{open message}, 0, m^0)$ and $(\texttt{open message}, 1, m^1)$ from $\mathcal{F}_{\mathsf{FCOT}}^n$ to all other parties.
6. **On output** $(\texttt{abort}, C')$ from all simulated parties, the simulator inputs $(\texttt{abort}, C')$ into the ideal $\mathcal{F}_{\mathsf{COM}}^n$.

The simulator provides an indistinguishable view:

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

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

Hybrid $\mathcal{H}_3$ In this game we modify the dummy functionality $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ 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 $\mathcal{F}_{\mathsf{FCOT}}^n$ and inputs $(\texttt{commit}, m)$ into $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$.
When the simulator receives $(\texttt{receipt commit})$ it let's the simulated sender input $(\texttt{messages}, 0, 0)$ into $\mathcal{F}_{\mathsf{FCOT}}^n$.
When the simulator receives $(\texttt{open}, m)$ from $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ it equivocates the simulated setup $\mathcal{F}_{\mathsf{FCOT}}^n$ 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 $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$'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 $\mathcal{F}_{\mathsf{FCOT}}^n$.

Hybrid $\mathcal{H}_4$ 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 $\mathcal{F}_{\mathsf{FCOT}}^n$ this never happens.

Hybrid $\mathcal{H}_5$ In this game we modify the dummy functionality to be $\mathcal{F}_{\mathsf{COM}}^n$, 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.

Hybrid $\mathcal{H}_6$ In this game the simulator has no formal abort condition. This is the ideal world execution. Note that the previous abort condition only happend 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 $\mathcal{F}_{\mathsf{FCOT}}^n$ at the cost of only committing to one bit.

### A.3 Commitment expansion

**Theorem 1.** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious. There exists a protocol $\pi_{\mathsf{COM}}$ in the $\{\mathcal{F}_{\mathsf{COM}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^n\}$-hybrid model that statistically UC-realizes $\mathcal{F}_{\mathsf{COM}}^n$:*

$$\{\mathcal{F}_{\mathsf{COM}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^n\} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}_{\mathsf{COM}}^n \tag{25}$$

*Proof.* Denote the set of parties by $P = \{\mathsf{S}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$, let $m \in \{0,1\}^\lambda$ be the message that the sender $\mathsf{S}$ commits to, let $\ell \coloneqq n\lambda^2$ be the size of the used secret sharing and let $\rho \coloneqq \lambda$ be the number of probed shares.

Formally, we posit our protocol in the $\{\mathcal{F}_{\mathsf{COM}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^n, \mathcal{F}_{\mathsf{CG}}^n\}$-hybrid model where $\mathcal{F}_{\mathsf{CG}}^n$ is realized using the broadcast as described in Appendix A.1. We require our protocol to be abort-respecting according to Definition 4 and argue that Rule 5 is fulfilled in the analysis.

Next we give the formal protocol where we use the notation $\mathcal{C}_l$ for the commitment setup $\mathcal{F}^{n-1}$ on parties $P \setminus \{\mathsf{R}_l\}$. After that we give more details on the protocol and a simulator to prove its security. For runtime restrictions, we assume that the unary security parameter $1^\lambda$ is also input.

Protocol $\pi_{\mathsf{COM}}$

1. **On input** $(\texttt{commit}, m \in \{0,1\}^\lambda)$ from $\mathcal{Z}$ to $\mathsf{S}$, the sender $\mathsf{S}$ creates a secret sharing $\mu \leftarrow \text{Share}_{\ell,2\ell}(m)$ and commits to each share by sending $(\texttt{commit}, (i, \mu_i))$ to $\mathcal{C}_l$ for all $l \in [n-1]$ and $i \in [2\ell]$.
2. **On output** $(\texttt{receipt commit})$ from $\mathcal{C}_l$ for all $l \in [n-1] \setminus \{j\}$, the receiver $\mathsf{R}_j$ broadcasts $(\texttt{probe}, \nu^j)$ where $\nu^j \leftarrow \binom{[2\ell]}{\rho}$ are uniformly random probing indices.
3. **On output** $(\texttt{output}, \mathsf{R}_j, (\texttt{probe}, \nu^j))$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $j \in [n-1]$, the sender opens all commitment setups $\mathcal{C}_l$ corresponding to the probed indices $\nu \coloneqq \bigcup_{i=1}^{n-1} \nu^j$.
4. **On output** $(\texttt{open}, (i, \mu_i^l))$ from $\mathcal{C}_l$ for all $l \neq j$ and $i \in \nu$, the receiver $\mathsf{R}_j$ broadcasts $(l, \mu_\nu^l)$.
5. **On output** $(\texttt{output}, \mathsf{R}_j, (l, \widetilde{\mu}_\nu^{l,j}))$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $j \in [n-1]$ and all $l \neq j$, the receiver $\mathsf{R}_k$ makes a list of *valid* setups (setups for which all receivers broadcasted the same shares).
Formally, $\mathsf{R}_k$ computes $V_{\text{probe}} \coloneqq \{(l, \widetilde{\mu}_\nu^l) \mid \forall j \neq l : \widetilde{\mu}^l = \widetilde{\mu}_\nu^{l,j}\}$.

Protocol $\pi_{\mathsf{COM}}$ (cont'd)

5.1. If at least one setup is valid ($|V_{\text{probe}}| \geq 1$) and the shares in all valid setups are consistent, i.e., $\exists \mu_\nu \in (\{0,1\}^\lambda)^\rho : \emptyset \neq V_{\text{probe}} \subseteq [n-1] \times \{\mu_\nu\}$, then $\mathsf{R}_j$ acknowledges the commitment by outputting (`commit receipt`).

5.2. If at least two setup are valid ($|V_{\text{probe}}| \geq 2$) and the valid messages are inconsistent, then each honest receiver aborts with $(\texttt{abort}, \mathsf{S})$.

5.3. If $|V_{\text{probe}}| = 0$, then $\mathsf{R}_j$ declares specific conflicts with each receiver who broadcasted a different message than $\mathsf{R}_j$ in any invalid setup. Specifically, for each invalid setup the honest parties proceed according to Rule 2 of Definition 4 to gradually biseparate the subgraph of the setup. Because all $n-1$ setups are invalid, all of their resp. subgraph are biseparated. Hence Lemma 6 applies and the overall Conflict Graph becomes biseparated. The honest parties abort accordingly.

6. **On input** (`open`) from $\mathcal{Z}$ to $\mathsf{S}$, the sender $\mathsf{S}$ sends (`open`) to $\mathcal{C}_l$ for all $l \in [n-1]$.

7. **On output** $(\texttt{open}, \mu_i^l)$ from $\mathcal{C}_l$ for all $l \neq j$ and $i \in [2\ell]$, the receiver $\mathsf{R}_j$ broadcasts $(l, \mu^l)$.

8. **On output** $(\texttt{output}, \mathsf{R}_j, (l, \widetilde{\mu}^{l,j}))$ from $\mathcal{F}_{\mathsf{BC}}^n$ for all $j \in [n-1]$ and $l \neq j$, the receiver $\mathsf{R}_k$ again makes a list of *valid* setups (setups for which all receivers broadcasted the same shares). Formally, $\mathsf{R}_k$ computes $V_{\text{open}} \coloneqq \{(l, \widetilde{\mu}^l) \mid \forall j \neq l : \widetilde{\mu}^l = \widetilde{\mu}^{l,j} \wedge (l, \widetilde{\mu}_\nu^l) \in V_{\text{probe}}\}$.

   8.1. If at least one setup is valid ($|V_{\text{open}}| \geq 1$) and the messages in all valid setups are consistent, i.e., $\exists \widetilde{\mu} \in (\{0,1\}^\lambda)^{2\ell} \cup \{\perp\} : \emptyset \neq V_{\text{open}} \subseteq [n-1] \times \{\widetilde{\mu}\}$, then
   - $\mathsf{R}_j$ outputs $(\texttt{open}, m)$, if $\perp \neq m \leftarrow \text{Recover}_{\ell,2\ell}(\widetilde{\mu})$,
   - otherwise $\mathsf{R}_j$ aborts with $(\texttt{abort}, \mathsf{S})$.

   8.2. If at least two setups are valid ($|V_{\text{open}}| \geq 2$) and the valid messages are inconsistent, then each honest receiver aborts with $(\texttt{abort}, \mathsf{S})$.

   8.3. If $|V_{\text{open}}| = 0$, then the honest parties proceed analogously to Step 5.3.

First, let's analyze the commitment phase. If all parties behave honestly, it is clear that each receiver obtains the correct message. However, there are three ways for malicious parties to disturb the protocol.

- The malicious receivers can claim to have received a different message from the opening of a setup commitment.
- The malicious sender can input different messages into the setups.
- The adversary can try to abort many setups to ensure that some receivers obtain on opened messages.

The validation step, i.e., computing $V_{\text{probe}}$ ensures that for each *valid* setup all $n$ parties agree on the opened shares. Because at least one honest party participates in each setup, the agreed open shares are equal to the actually opened shares. Consequently, for each *valid* setup, all parties (even outside the setup) learn the shares that were opened.

With this intuition in mind, proving security of this protocol is straightforward. We start by describing a simulator that has access to the ideal $\mathcal{F}_{\mathsf{COM}}^n$ and simulates the honest parties and the setups (hybrid functionalities). In particular, the dummy adversary's and the corrupted parties' local in- and output, as well as, messages to and from (simulated) setups are forwarded from and to the environment.

Simulator for $\pi_{\mathsf{COM}}$

1. **On output** (`receipt commit`) from $\mathcal{F}_{\mathsf{COM}}^n$, the simulator gives local input $(\texttt{commit}, 0)$ to the honest simulated sender who then inputs $(\texttt{commit}, \mu)$ into all (simulated) $\mathcal{C}_l$ where $\mu \leftarrow \text{Share}_{\ell,2\ell}(0)$.
2. **On output** (`receipt commit`) from all simulated receivers, at this point, the malicious sender must already have provided the simulated setups $\mathcal{C}_l$ with input $(\texttt{commit}, \mu^l)$. The simulator reconstructs $m^l \leftarrow \text{Recover}_{\ell,2\ell}(\mu^l)$ for each *valid* setup, i.e., each setup $\mathcal{C}_l$ s.t. $(l, \cdot) \in V_{\text{probe}}$. Then the simulator chooses the smallest $\hat{l}$ s.t. $\mathcal{F}_{\hat{l}}^{n-1}$ is valid and $m^{\hat{l}} \neq \perp$, and inputs $(\texttt{commit}, m^{\hat{l}})$ into the ideal $\mathcal{F}_{\mathsf{COM}}^n$ in the name of the (malicious) sender. **(Extraction)**
   If no such $\hat{l}$ exists, the simulator inputs 0 instead.
   Note that, due to the protocol's probing step, with overwhelming probability the encoded messages of all valid setup are equal, i.e., it does not matter which one the simulator chooses.
3. **On output** $(\texttt{open}, m)$ from $\mathcal{F}_{\mathsf{COM}}^n$, the simulator lets the (honest) simulated sender open the simulated setups. However, the simulator equivocates the setups to a sharing that encodes $m$ (which $\mathcal{S}$ just learned).
   More precisely, the simulator creates a random sharing $\mu'$ s.t. $m \leftarrow \text{Recover}_{\ell,2\ell}(\mu')$ and the sharing matches the already probed shares $\mu'_\nu = \mu_\nu$. This is possible because changing the encoded message requires changing the sharing on $\ell$ many indices, while the simulator can manipulate $2\ell - n\rho = 2n\lambda^2 - n\lambda > n\lambda^2 = \ell$ (non-opened) indices. **(Equivocation)**
4. **On output** $(\texttt{open}, m)$ from all simulated receivers, the simulator inputs $(\texttt{open})$ into the ideal $\mathcal{F}_{\mathsf{COM}}^n$ in the name of the (malicious) sender.
5. **On output** $(\texttt{abort}, C')$ from all simulated parties, the simulator inputs $(\texttt{abort}, C')$ into the ideal $\mathcal{F}_{\mathsf{COM}}^n$.

We now provide a formal proof that this protocol indeed instantiates $\mathcal{F}_{\mathsf{COM}}^n$. To that end we follow the traditional approach and start in a setting where each of the $n$ parties executes the protocol and successively change minor details until we end up in the ideal world where honest parties directly forward their input into $\mathcal{F}_{\mathsf{COM}}^n$ and the simulator reports messages for the honest parties by executing the code above. By showing (statistical) indistinguishability between each two consecutive games we prove the indistinguishability of the real and ideal world execution.

Hybrid $\mathcal{H}_1$ This game corresponds to the real world where all parties execute the protocol honestly.

Hybrid $\mathcal{H}_2$ This game introduces an ideal dummy functionality $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ that does nothing but relaying the information between the honest sender and the simulator who runs the real protocol in its head. Honest parties are replaced by dummy parties. The simulated parties' outputs are relayed to the $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ which relays them to the dummy parties and finally to the environment.
In this game the same code is executed, the only difference is that the messages are relayed through the dummy functionality and the dummy parties.

Hybrid $\mathcal{H}_3$ This game is the same as the previous one except that if the sender $\mathsf{S}$ is honest, the simulator $\mathcal{S}$ on relayed input $(\texttt{commit}, m)$ from $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ for $\mathsf{S}$ gives the simulated (honest) sender local input $(\texttt{commit}, 0)$ instead. Eventually, the honest (simulated) parties output (`receipt commit`) which is relayed by $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ to the dummy parties. After input $(\texttt{open})$, when the simulated sender opens the simulated commitment setups $\mathcal{C}_l$ the simulator equivocates the (non-probed) shares to a sharing that encodes $m$. Finally, the honest (simulated) parties output $(\texttt{open}, m)$ which is relayed through $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ and the dummy parties.
This is possible because changing the encoded message of a sharing requires manipulating $\ell+1$ many indices while the simulator is at liberty to manipulate at least $2\ell - n\rho = 2n\lambda^2 - n\lambda > n\lambda^2 = \ell$ (non-probed) shares.

Hybrid $\mathcal{H}_4$ This game is the same as the previous one except that if the sender is honest, then the ideal functionality $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ acts as the ideal functionality $\mathcal{F}_{\mathsf{COM}}^n$. On input $(\texttt{commit}, m)$ from the dummy sender the ideal functionality $\widetilde{\mathcal{F}}_{\mathsf{COM}}^n$ no longer forwards $(\texttt{commit}, m)$ to the simulated sender but instead sends (`receipt commit`) to all dummy parties and the simulator. As in the previous game the simulator gives local input $(\texttt{commit}, 0)$ the simulated (honest) sender but does not forward the simulated parties' outputs. (Recall the dummy parties already output (`receipt commit`) to the environment, just as the honest parties would in the real protocol.)

On input (open) from the dummy sender, the ideal functionality $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$ outputs $(\texttt{open}, m)$ to all dummy parties and the simulator.
The outputs of the dummy parties in this game are the same as in the previous one, the only difference is that now the outputs are initiated by $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$ instead of being initiated by the simulated protocol and relayed through $\widetilde{\mathcal{F}}^n_{\mathsf{COM}}$.

Hybrid $\mathcal{H}_5$ This game is the same as the previous one except that if the sender $\mathsf{S}$ is corrupted, the simulator $\mathcal{S}$, on output (receipt commit) from all simulated parties, extracts a message $m'$ from the corrupted sender's inputs into the commitment setups $\mathcal{C}_l$. Concretely, the simulator reconstructs $m^l \leftarrow \text{Recover}_{\ell,2\ell}(\mu^l)$ for each *valid* setup. Then $\mathcal{S}$ finds the minimal $\hat{l}$ s.t. $m^{\hat{l}} \neq \perp$ and $\mathcal{C}_{\hat{l}}$ is valid. At the end of the opening phase, $\mathcal{S}$ aborts iff $m'$ differs from any of the outputs of the simulated parties $(\texttt{open}, m)$.
The simulator's abort probability[13] is negligible. This is ensured by the protocol's probing step: first we see that—upon outputting a receipt—all commitment setups contain sharings that encode the same message $m$, or $\perp$. To see this, suppose there were two setups $\mathcal{C}_{l_1}$ and $\mathcal{C}_{l_2}$ whose committed sharings $\mu^1$ and $\mu^2$ encode different messages $m^1 = \text{Recover}_{\ell,2\ell}(\mu^1) \neq \perp$ and $m^2 = \text{Recover}_{\ell,2\ell}(\mu^2) \neq \perp$. Then these two sharings must differ on at least $\ell$ indices $W \in \binom{[2\ell]}{\ell}$. However, since at least two parties are honest, at least $2\rho$ probing shares are uniformly random. Hence, the probability that none of the $2\rho$ uniform probing shares hit any such differing index $W$ is bounded by Lemma 1

$$\Pr_{\nu \leftarrow \binom{[2\ell]}{\rho}}[W \cap \nu = \emptyset] \leq 2^{-2\rho\ell/2\ell} = 2^{-\rho} = 2^{-\lambda} \ . \tag{31}$$

Via a union bound we find that the probability of any two setups (out of $n$) containing different encoded messages (not $\perp$) is at most $n^2 2^{-\lambda}$. Conversely, if the honest parties output a receipt for the commitment, we have with overwhelming probability that all setups contain the same encoded message or $\perp$. Lastly, we argue that if all setups contain the same encoded message $m$, then any honest (simulated) party will eventually output that message (if the protocol is not aborted). Hence, the simulator's abort probability is exponentially small.

Hybrid $\mathcal{H}_6$ This game is the same as the previous one except that the ideal functionality is now $\mathcal{F}^n_{\mathsf{COM}}$ (independent of whether $\mathsf{S}$ is corrupt). Moreover, if the sender $\mathsf{S}$ is corrupted, the simulator $\mathcal{S}$, on output (receipt commit) from all simulated parties, gives input $(\texttt{commit}, m')$ to $\mathcal{F}^n_{\mathsf{COM}}$ in the name of the (dummy) sender. Similarly, on output $(\texttt{open}, m)$ from all simulated parties, the simulator gives input (open) to $\mathcal{F}^n_{\mathsf{COM}}$ in the name of the dummy sender.
Again, the dummy parties' outputs are the same as in the previous game. If the simulator does not abort (if $m' = m$), the dummy parties' outputs are $(\texttt{open}, m)$ in this game and $(\texttt{open}, m')$ in the previous one. If the simulator aborts, then the dummy parties' output are also the same, i.e., non-existent.

Hybrid $\mathcal{H}_7$ This game is the same as the previous one except that the simulator has no formal abort condition. Since the abort only happens with negligible probability anyways the output distribution of this game is statistical close to the previous one. Note that this is the ideal world execution.

□

### A.4 FCOT expansion

**Theorem 2.** *Let $n \geq 3$ be the number of parties of which at most $0 \leq t \leq n-3$ are malicious. There exists a protocol $\pi_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^{n-1}_{\mathsf{FCOT}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model that statistically UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$:*

$$\{\mathcal{F}^{n-1}_{\mathsf{FCOT}}, \mathcal{F}^n_{\mathsf{BC}}\} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}^n_{\mathsf{FCOT}} \tag{26}$$

*Proof.* Denote the set of parties by $P = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, ..., \mathsf{W}_{n-2}\}$, let $m^0, m^1 \in \{0,1\}^\lambda$ be the sender's messages, let $c \in \{0,1\}$ be the receiver's choice bit, let $\ell := n\lambda^2$ be the size of the used secret sharing and let $\rho := \lambda$ be the number of probed shares.

Formally, we posit our protocol in the $\{\mathcal{F}^{n-1}_{\mathsf{SFE},f}, \mathcal{F}^n_{\mathsf{BC}}, \mathcal{F}^n_{\mathsf{CG}}\}$-hybrid model where $\mathcal{F}^n_{\mathsf{CG}}$ is realized using the broadcast as described in Appendix A.1, and $\mathcal{V}_\kappa$ is realized using calls to $\mathcal{F}^{n-1}_{\mathsf{FCOT}}$ via Lemma 9. We require our protocol to be abort-respecting according to Definition 4 and argue that

[13] Note that this abort has nothing to do with the *Identifiable Abort* for the ideal functionality.

Rule 5 is fulfilled in the analysis. In particular, we denote by $\mathcal{V}_\kappa$ the functionality $\mathcal{F}^{n-1}_{\mathsf{SFE},f=f_{\mathrm{OT}}}$ on parties $P \setminus \mathsf{W}_\kappa$ evaluates the following (randomized) function

$$f_{\mathrm{OT}} : \begin{pmatrix} x_{\mathsf{S}} = (\mu^{0,\hat{\kappa}}, \mu^{1,\hat{\kappa}}, \eta^{0,\hat{\kappa}}, \eta^{1,\hat{\kappa}})_{\hat{\kappa}\in[n-1]} \\ x_{\mathsf{R}} = (\gamma^{\hat{\kappa}})_{\hat{\kappa}\in[n-1]} \\ x_{\mathsf{W}_1} = \varepsilon \\ \vdots \end{pmatrix} \mapsto \begin{pmatrix} y_{\mathsf{S}} = (\nu^\kappa, \gamma^\kappa_{\nu^\kappa}) \\ y_{\mathsf{R}} = (\nu^\kappa, \mu^{c^\kappa,\kappa}_{\nu^\kappa} \mu^{1-c^\kappa,\kappa}_{\nu^\kappa}, \eta^{0,\kappa}_{\nu^\kappa}, \eta^{1,\kappa}_{\nu^\kappa}) \\ y_{\mathsf{W}_1} = (\nu^\kappa, \mu^{0,\kappa}_{\nu^\kappa}, \mu^{1,\kappa}_{\nu^\kappa}, \eta^{0,\kappa}_{\nu^\kappa}, \eta^{1,\kappa}_{\nu^\kappa}, \gamma^\kappa_{\nu^\kappa}) \\ \vdots \end{pmatrix} \tag{32}$$

where $\nu^\kappa \leftarrow \binom{[2\ell]}{\rho}$ is uniformly random. The function $f_{\mathrm{OT}}$ takes as inputs $n-1$ times two sharings $\mu^{0,\kappa}, \mu^{1,\kappa}$ (of the sender's messages) and $n-1$ times a sharing $\gamma^\kappa$ (of the receiver's choice). It verifies that the sharings encode the same message

$$\forall b \in \{0,1\} : \exists m^b \neq \bot : \forall \kappa \in [n-1] : m^b \stackrel{?}{=} m^{b,\kappa} \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu^{b,\kappa}) \oplus \mathrm{Recover}_{\ell,2\ell}(\eta^{b,\kappa}) \tag{33}$$

and the same choice bit

$$\exists c \neq \bot : \forall \kappa \in [n-1] : c \stackrel{?}{=} c^\kappa \leftarrow \mathrm{Recover}_{\ell,2\ell}(\gamma^\kappa) \ . \tag{34}$$

If the verification succeeds $f_{\mathrm{OT}}$ samples a set of indices used for probing the shares $\nu^\kappa$. This set, alongside all corresponding shares, are output to all parties. If the verification fails, the resp. inputs are ignored.

Next, we give the formal protocol followed by more details on the protocol and a simulator to prove its security. For runtime restrictions, we assume that the unary security parameter $1^\lambda$ is also input.

Protocol $\pi_{\mathsf{FCOT}}$

1. **On input** $(\mathtt{messages}, m^0, m^1 \in \{0,1\}^\lambda)$ from $\mathcal{Z}$ to $\mathsf{S}$, the sender $\mathsf{S}$ samples masks $w^{0,\kappa}, w^{1,\kappa} \leftarrow \{0,1\}^{n-2}$ and creates secret sharings $\mu^{b,\kappa} \leftarrow \mathrm{Share}_{\ell,2\ell}(m^b \oplus w^{b,\kappa})$ and $\eta^{b,\kappa} \leftarrow \mathrm{Share}_{\ell,2\ell}(w^{b,\kappa})$ for each $b \in \{0,1\}$ and each $\kappa \in [n-1]$ independently. The sender commits to each mask by sending $(\mathtt{commit}, (b, \kappa, w^{b,\kappa}))$ and to each share by sending $(\mathtt{commit}, (b, \kappa, i, \mu^{b,\kappa}_i))$ resp. $(\mathtt{commit}, (b, \kappa, i, \eta^{b,\kappa}_i))$ to $\mathcal{F}^n_{\mathsf{COM}}$ for each $b \in \{0,1\}$, $\kappa \in [n-1]$ and $i \in [2\ell]$. Finally, the sender inputs its sharings $(\mu^{0,\hat{\kappa}}, \mu^{1,\hat{\kappa}}, \eta^{0,\hat{\kappa}}, \eta^{1,\hat{\kappa}})_{\hat{\kappa}\in[n-1]}$ into $\mathcal{V}_\kappa$ for each $\kappa \in [n-1]$.
2. **On input** $(\mathtt{choice}, c \in \{0,1\})$ from $\mathcal{Z}$ to $\mathsf{R}$, the receiver $\mathsf{R}$ creates $n-1$ secret sharings $\gamma^\kappa \leftarrow \mathrm{Share}_{\ell,2\ell}(c)$. The sender commits to each share by sending $(\mathtt{commit}, (i, \gamma^\kappa_i))$ to $\mathcal{F}^n_{\mathsf{COM}}$ for each $\kappa \in [n-1]$ and $i \in [2\ell]$. Finally, the receiver inputs its sharings $(\gamma^{\hat{\kappa}})_{\hat{\kappa}\in[n-1]}$ into $\mathcal{V}_\kappa$ for each $\kappa \in [n-1]$.
3. **On output** $(\mathtt{output}, y_{\mathsf{S}})$ from $\mathcal{V}_\kappa$ to $\mathsf{S}$, the sender $\mathsf{S}$ opens its global commitments corresponding to all shares with indices $\nu^\kappa$ and broadcasts $(\kappa, \nu^\kappa, (\mu^{0,\hat{\kappa}}_{\nu^\kappa}, \mu^{1,\hat{\kappa}}_{\nu^\kappa}, \eta^{0,\hat{\kappa}}_{\nu^\kappa}, \eta^{1,\hat{\kappa}}_{\nu^\kappa})_{\hat{\kappa}\in[n-1]})$.
4. **On output** $(\mathtt{output}, y_{\mathsf{R}})$ to $\mathsf{R}$, the receiver $\mathsf{R}$ opens its global commitments corresponding to all shares with indices $\nu^\kappa$ and broadcasts $(\kappa, \nu^\kappa, (\gamma^{\hat{\kappa}}_{\nu^\kappa})_{\hat{\kappa}\in[n-1]})$.
5. **On output** $(\mathtt{output}, y^\kappa_{\mathsf{W}_{\kappa'}})$ from $\mathcal{V}_\kappa$ to $\mathsf{W}_{\kappa'\neq\kappa}$, the witness $\mathsf{W}_{\kappa'}$ stores $y^\kappa_{\mathsf{W}_{\kappa'}}$.
6. **On output** $(\mathtt{output}, \mathsf{S}, (\kappa, \nu^\kappa, (\mu^{0,\hat{\kappa}}_{\nu^\kappa}, \mu^{1,\hat{\kappa}}_{\nu^\kappa}, \eta^{0,\hat{\kappa}}_{\nu^\kappa}, \eta^{1,\hat{\kappa}}_{\nu^\kappa})_{\hat{\kappa}\in[n-1]}))$ from $\mathcal{F}^n_{\mathsf{BC}}$, any party $\mathsf{P}$ checks if the broadcasted shares match the probed shares from the $\mathcal{V}_\kappa$. If not, the party $\mathsf{S}$ must be malicious; the participating parties biseparate the Conflict Graph on parties $P \setminus \{\mathsf{W}_\kappa\}$ according to Rule 2 of Definition 4. Effectively, $\mathcal{V}_\kappa$ is aborted.
7. **On output** $(\mathtt{output}, \mathsf{R}, (\kappa, \nu^\kappa, (\gamma^{\hat{\kappa}}_{\nu^\kappa})_{\hat{\kappa}\in[n-1]}))$ from $\mathcal{F}^n_{\mathsf{BC}}$, any party $\mathsf{P}$ checks if the broadcasted shares match the probed shares from the $\mathcal{V}_\kappa$. If not, the party $\mathsf{R}$ must be malicious; the participating parties biseparate the Conflict Graph on parties $P \setminus \{\mathsf{W}_\kappa\}$ according to Rule 2 of Definition 4. Effectively, $\mathcal{V}_\kappa$ is aborted.
8. **On output** $(\mathtt{open}, (b, \kappa, i, \mu^{b,\kappa}_i))$ from $\mathcal{F}^n_{\mathsf{COM}}$ for all $i \in \nu^\kappa$, any party $\mathsf{P} \neq \mathsf{W}_\kappa$ checks if the opened shares match the broadcasted (and agreed upon) ones for $\mathcal{V}_\kappa$. If the shares are inconsistent, then the honest parties accuse the malicious sender and apply Rule 2 of Definition 4 to eventually biseparate the subgraph $P \setminus \{\mathsf{W}_\kappa\}$. Effectively, $\mathcal{V}_\kappa$ is aborted.
9. **On output** $(\mathtt{open}, (b, \kappa, i, \gamma^\kappa_i))$ from $\mathcal{F}^n_{\mathsf{COM}}$ for all $i \in \nu^\kappa$, any party $\mathsf{P} \neq \mathsf{W}_\kappa$ checks if the opened shares match the broadcasted (and agreed upon) ones for $\mathcal{V}_\kappa$. If the shares are inconsistent, then the honest parties accuse the malicious receiver and apply Rule 2 of Definition 4

Protocol $\pi_{\mathsf{FCOT}}$ (cont'd)

to eventually biseparate the subgraph $P \setminus \{\mathsf{W}_\kappa\}$. Effectively, $\mathcal{V}_\kappa$ is aborted.
If all shares (the sender's and the receiver's) are consistent, then the sender opens the mask sharings $\eta^{0,\widetilde{\kappa}}$ and $\eta^{1,\widetilde{\kappa}}$ for the smallest (non-aborted) setup $\mathcal{V}_{\widetilde{\kappa}}$.

10. **On output** $(\texttt{open}, (b, \widetilde{\kappa}, \eta^{b,\widetilde{\kappa}}))$ from $\mathcal{F}_{\mathsf{COM}}^n$ for the correct $\widetilde{\kappa}$, the receiver computes $w^{c,\widetilde{\kappa}} \leftarrow \mathrm{Recover}_{\ell,2\ell}(\eta^{c,\widetilde{\kappa}}) \oplus w^{c,\widetilde{\kappa}}$ and $m^c \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu^{c,\widetilde{\kappa}}) \oplus w^{c,\widetilde{\kappa}}$, and outputs $(\texttt{output}, m^c)$. Each other party outputs $(\texttt{receipt transfer})$ and stores $\widetilde{\kappa}$.
11. **On input** $(\texttt{open message}, b \in \{0,1\})$ from $\mathcal{Z}$ to $\mathsf{S}$, the sender $\mathsf{S}$ opens the commitments to all its shares corresponding to the sharing $\mu^{b,\widetilde{\kappa}}$.
12. **On output** $(\texttt{open}, (b, \widetilde{\kappa}, i, \mu_i^{b,\widetilde{\kappa}}))$ from $\mathcal{F}_{\mathsf{COM}}^n$ for $i \in [2\ell]$ and $b \in \{0,1\}$, each party reconstructs $m^b \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu^{b,\widetilde{\kappa}}) \oplus w^{b,\widetilde{\kappa}}$ where $w^{b,\widetilde{\kappa}}$ is already known from the commitment phase. Then each party $\mathsf{P} \neq \mathsf{S}$ outputs $(\texttt{open message}, b, m^b)$.
13. **On input** $(\texttt{open choice})$ from $\mathcal{Z}$ to $\mathsf{R}$, the receiver $\mathsf{R}$ opens the commitments to all its shares.
14. **On output** $(\texttt{open}, (b, \widetilde{\kappa}, i, \mu_i^{b,\widetilde{\kappa}}))$ from $\mathcal{F}_{\mathsf{COM}}^n$ for all $i \in [2\ell]$ and $b \in \{0,1\}$, each party reconstructs $c \leftarrow \mathrm{Recover}_{\ell,2\ell}(\gamma^{\widetilde{\kappa}})$. Then each party $\mathsf{P} \neq \mathsf{R}$ outputs $(\texttt{open choice}, c)$.

On a high level, the protocol lets the sender commit globally to its masked messages, in the form of a secret-sharing, and to sharings of its masks. The receiver also globally commits to a sharing of its choice bit. We use a probing mechanism that works as follows: both the sender and the receiver *commit* to all shares using $\mathcal{F}_{\mathsf{COM}}^n$ and insert the same shares into the setups $\mathcal{V}_\kappa$. The setup $\mathcal{V}_\kappa$ then samples a set $\nu^\kappa$ uniformly at random which contains indices of probed shares. This set is output to all parties alongside the shares for both sender inputs and the receiver inputs at these indices. This way, all parties know which values were input into $\mathcal{V}_\kappa$ at those indices. By unveiling the global commitments at those indices the sender and the receiver can prove that their globally committed sharings and their sharings in $\mathcal{V}_\kappa$ encode the same values.

Through the setups the receiver learns the selected message, but information-theoretically hidden by the mask $w^c$. Because the sender only opens one pair of masks, the receiver can also only learn the chosen message from one setup $\mathcal{V}_{\widetilde{\kappa}}$. To open, each party simply opens its global commitments, so that all parties can verify their sharing and reconstruct the resp. value.

We now provide a simulator that provides indistinguishable transcript for any environment $\mathcal{Z}$.

Simulator for $\pi_{\mathsf{FCOT}}$

1. **On output** $(\texttt{receipt transfer})$ from $\mathcal{F}_{\mathsf{FCOT}}^n$: if the sender is honest, $\mathcal{S}$ gives local input $(\texttt{message}, 0, 0)$ to the honest simulated sender.
   If the receiver is honest, $\mathcal{S}$ gives local input $(\texttt{choice}, 0)$ to the honest simulated receiver.
2. **On output** $(\texttt{receipt transfer})$ from all simulated parties: if the sender is corrupted, $\mathcal{S}$ finds the index $\widetilde{\kappa}$, i.e., the setups for which the sender's masks $w^{0,\widetilde{\kappa}}, w^{1,\widetilde{\kappa}}$ are opened. Then the simulator extracts the sharings $\mu^{0,\widetilde{\kappa}}, \mu^{1,\widetilde{\kappa}}, \eta^{0,\widetilde{\kappa}}, \eta^{1,\widetilde{\kappa}}$ from the sender's input to $\mathcal{F}_{\mathsf{COM}}^n$. The simulator reconstructs $m^0 \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu^{0,\widetilde{\kappa}}) \oplus \mathrm{Recover}_{\ell,2\ell}(\eta^{0,\widetilde{\kappa}})$ and $m^1 \leftarrow \mathrm{Recover}_{\ell,2\ell}(\mu^{1,\widetilde{\kappa}}) \oplus \mathrm{Recover}_{\ell,2\ell}(\eta^{1,\widetilde{\kappa}})$ and inputs $(\texttt{messages}, m^0, m^1)$ into the ideal $\mathcal{F}_{\mathsf{FCOT}}^n$ in the name of the sender. **(Extraction)**
   If the receiver is corrupted, $\mathcal{S}$ finds the index $\widetilde{\kappa}$, i.e., the setups for which the sender's masks are opened. Then the simulator extracts the sharing $\gamma^{\widetilde{\kappa}}$ from the receiver's input to $\mathcal{F}_{\mathsf{COM}}^n$. The simulator reconstructs $c \leftarrow \mathrm{Recover}_{\ell,2\ell}(\gamma^{\widetilde{\kappa}})$ and inputs $(\texttt{choice}, c)$ into the ideal $\mathcal{F}_{\mathsf{FCOT}}^n$ in the name of the receiver. **(Extraction)**
3. **On output** $(\texttt{open message}, b, m^b)$ from $\mathcal{F}_{\mathsf{FCOT}}^n$, the simulator lets the (honest) simulated sender open the simulated global commitments. However, the simulator equivocates the global setups to shares that encode $m^b$ (which $\mathcal{S}$ just learned). This step is analogous to the simulator's Step 3 in the proof of Theorem 1. Here, at most $n\rho$ shares are probed, leaving $2\ell - n\rho = 2n\lambda^2 - n\lambda > n\lambda^2 = \ell$ many shares to be equivocated. **(Equivocation)**
4. **On output** $(\texttt{open message}, b, m^b)$ from all simulated parties (expect $\mathsf{S}$), the simulator inputs $(\texttt{open message}, b)$ into $\mathcal{F}_{\mathsf{COM}}^n$ in the name of the sender.

Simulator for $\pi_{\mathsf{FCOT}}$ (cont'd)

5. **On output** $(\texttt{open choice}, c)$ from $\mathcal{F}_{\mathsf{FCOT}}^n$, the simulator lets the (honest) simulated receiver open the simulated global commitments. However, the simulator equivocates the global setups to shares that encode $m^0$ resp. $m^1$ (which $\mathcal{S}$ just learned). This step is analogous to the previous Step 3. **(Equivocation)**
   **On output** $(\texttt{open choice}, c)$ from all simulated parties (expect $\mathsf{R}$), the simulator inputs $(\texttt{open choice})$ into $\mathcal{F}_{\mathsf{COM}}^n$ in the name of the receiver.
6. **On output** $(\texttt{abort}, C')$ from all simulated parties, the simulator inputs $(\texttt{abort}, C')$ into the ideal $\mathcal{F}_{\mathsf{FCOT}}^n$.

We will now proceed similarly to the proof of Theorem 1 and prove that the simulator indeed provides an indistinguishable transcript for any environment $\mathcal{Z}$. The games we use to prove our claim are as follows:

Hybrid $\mathcal{H}_1$ This game corresponds to the real world. That is, all honest parties execute the protocol according to their code, and all corrupted parties are played by the dummy adversary who is controlled by the environment $\mathcal{Z}$.

Hybrid $\mathcal{H}_2$ This game introduces an ideal dummy functionality $\widetilde{\mathcal{F}}_{\mathsf{FCOT}}^n$ that does nothing but relaying the information between the honest sender and the simulator who runs the real protocol in its head. Honest parties are replaced by dummy parties. The simulated parties' outputs are relayed to the $\widetilde{\mathcal{F}}_{\mathsf{FCOT}}^n$ which relays them to the dummy parties and finally to the environment.
In this game the same code is executed, the only difference is that the messages are relayed through the dummy functionality and the dummy parties.

Hybrid $\mathcal{H}_3$ This game is the same as the previous one except that for a corrupted sender (resp. receiver) the simulator extracts its input $\hat{m}^0, \hat{m}^1$ (resp. $\hat{c}$) and aborts if it does not match the output of the simulated parties. More precisely, the simulator takes the messages the corrupted sender has input into $\mathcal{F}_{\mathsf{COM}}^n$ and checks if the encoded messages are consistent: for $b \in \{0,1\}^*$ and $\kappa \in [n-1]$ the simulator reconstructs the mask $w^{b,\kappa}$ as $\mathrm{Recover}_{\ell,2\ell}(\eta^{b,\kappa})$ and then the message $m^{b,\kappa} \coloneqq \mathrm{Recover}_{\ell,2\ell}(\mu^{b,\kappa}) \oplus w^{b,\kappa}$. Now let

$$M^b \coloneqq \left\{ m^{b,\kappa} \;\middle|\; \kappa \in [n-1] \right\} . \tag{35}$$

If there exists some $a \in M^b \cap \{0,1\}^\lambda$ such that $M^b \subseteq \{a, \perp\}$, meaning that all successful reconstructions yield the same value $a$ and at least one reconstruction has been successful, the simulator sets $\hat{m}^b \coloneqq a$. Otherwise, the simulator samples a uniformly random $\hat{m}^b \xleftarrow{\$} \{0,1\}^\lambda$. Analogously, let $c^\kappa \coloneqq \mathrm{Recover}_{\ell,2\ell}(\gamma^\kappa)$ and let

$$C \coloneqq \{c^\kappa \mid \kappa \in [n-1]\} . \tag{36}$$

The simulator sets $\hat{c} \coloneqq a$ if there exists some $a \in C \cap \{0,1\}$ such that $C \subseteq \{a, \perp\}$, and $\hat{c} \leftarrow \{0,1\}$ otherwise.
At the end of the protocol, the simulator aborts if $\hat{m}^0, \hat{m}^1$ and $\hat{c}$ do not match the outputs of the resp. parties, i.e., $(\texttt{output}, \hat{m}^{\hat{c}})$ for the receiver after the OT, $(\texttt{open message}, b, \hat{m}^b)$ for the sender and $(\texttt{open choice}, \hat{c})$ for the receiver when opening.
Now, we show that the probability of abort is negligible. As specified by the protocol, when the receiver outputs $(\texttt{output}, m^c)$ then the sharings inputs into $\mathcal{F}_{\mathsf{COM}}^n$ and the sharings input into any $\mathcal{V}_\kappa$ setup must encode the same message (or $\perp$) with overwhelming probability. This follows from the fact that $\rho$ many shares of each sharings are probed, thus sufficiently many $(\ell+1)$ differing sharings are detected with overwhelming probability. As we give a formal analysis in Hybrid $\mathcal{H}_5$ in the proof of Theorem 1, we omit the formulas here.

Hybrid $\mathcal{H}_4$ This game is the same as the previous one except the ideal functionality $\widetilde{\mathcal{F}}_{\mathsf{FCOT}}^n$ now acts as the ideal functionality $\mathcal{F}_{\mathsf{FCOT}}^n$. On input $(\texttt{messages}, m^0, m^1)$ (resp. $(\texttt{choice}, c)$) from the dummy sender (resp. receiver) the ideal functionality no longer forwards the input to the simulated party but instead sends $(\texttt{receipt transfer})$ to all dummy parties and the simulator, and $(\texttt{output}, m^c)$ to the dummy receiver.

- If the sender is honest and the receiver is honest, the simulator then gives input $(\texttt{messages}, 0, 0)$ to the simulated sender and $(\texttt{choice}, 0)$ to the simulated receiver.
- If the sender is honest and the receiver is corrupted, the simulator extracts the receiver's choice bit $\hat{c}$ as in the previous game. The simulator then inputs $(\texttt{choice}, \hat{c})$ into $\mathcal{F}_{\mathsf{FCOT}}^n$ in

the name of $\mathsf{R}$ to obtain $(\texttt{output}, m^{\hat{c}})$. The simulator then gives input $(\texttt{messages}, \widetilde{m}^0, \widetilde{m}^1)$ to the simulated sender where $\widetilde{m}^{\hat{c}} := m^{\hat{c}}$ and $\widetilde{m}^{1-\hat{c}} := 0^\lambda$.

- If the sender is corrupted and the receiver is honest, the simulator extracts the sender's messages $\hat{m}^0, \hat{m}^1$ as in the previous game. The simulator then inputs $(\texttt{messages}, \hat{m}^0, \hat{m}^1)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$ in the name of $\mathsf{S}$. The simulator then gives input $(\texttt{choice}, 0)$ to the simulated receiver.
- If the sender is corrupted and the receiver is corrupted, the simulator extracts both the sender's messages and the receiver's choice bit, inputs them into $\mathcal{F}^n_{\mathsf{FCOT}}$ and gives the appropriate local input to the simulated parties.

Furthermore, in the simulated protocol run the simulator equivocates committed sharings to match the outputs from $\mathcal{F}^n_{\mathsf{FCOT}}$. Concretely, on $(\texttt{open message}, b, m^b)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ the simulator equivocates the (simulated) sender's mask sharing $\eta^{b,\hat{\kappa}}$ to sharings that encode $m^b$. This is possible because changing the encoded value in the sharings requires changing $\ell + 1$ shares where the simulator can equivocate $2\ell - n\rho > \ell$ (non-probed) indices per sharing. This is the same argument as in Hybrid $\mathcal{H}_3$ in the proof of Theorem 1.

Note that, conditioned on the simulator not aborting, the simulated parties output is the same as before due to the simulator's extraction and subsequent equivocation. Hence this game's output distribution is statistically close to the previous one.

Hybrid $\mathcal{H}_5$ This game is the same as the one before except that the simulator has no formal abort condition. Since the abort in this game never happens, the output distribution of this game is the same as the previous one.

Note that Hybrid $\mathcal{H}_5$ corresponds to the ideal world where the simulator executes the code above, the functionality acts as specified and all honest parties are dummies. The proof was largely simplified by the fact that witnesses have no secrets and hence the simulator can just follow their protocol, whereas both sender and receiver have secrets which can be directly extracted from the setups or equivocated by adjusting the shares that were not probed.

Finally, analyzing the Identifiable Abort property we can use Lemma 6 to ensure that at least one $\mathcal{V}_\kappa$ setup cannot be aborted without biseparating the overall Conflict Graph. Note that there are $n - 2 > n - 3 \geq t$ such setups $\mathcal{V}_\kappa$, hence Lemma 6 applies, if all such setups are aborted. □

### A.5 Equivalence of FCOT, SFE and Correlated-Randomness

In this section we prove the three constructions in

**Lemma 7.** *For every number of parties $n$, the functionalities $\mathcal{F}^n_{\mathsf{SFE},f}$, $\mathcal{F}^n_{\mathsf{FCOT}}$ and $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$ are equally powerful.*

$$\mathcal{F}^n_{\mathsf{FCOT}} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}^n_{\mathsf{SFE},f} \overset{\text{perf}}{\rightsquigarrow} \mathcal{F}^n_{\mathsf{Corr},\mathcal{D}} \overset{\text{stat}}{\rightsquigarrow} \mathcal{F}^n_{\mathsf{FCOT}} \tag{27}$$

***The IPS-compiler.*** Before showing the construction $\mathcal{F}^n_{\mathsf{FCOT}} \rightsquigarrow \mathcal{F}^n_{\mathsf{SFE},f}$ we first recall the so-called IPS-compiler [37] (see also [41] 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 $\Pi$ provides security against a malicious adversary but only for an honest majority (that is, $t < n/2$), while the inner protocol $\rho$ is secure against arbitrarily many semi-honest parties. The resulting protocol $\Phi$ 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 $\Phi$ that lets them use the outer protocol $\Pi$ to additively share their input with the $m$ virtual parties, who then execute the inner protocol $\rho$ 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 $\Pi$ making black-box use of the inner

protocol $\rho$ 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 $\mathsf{P}$ that is in possession of the one-time pad that another client $\mathsf{P}'$ uses for the $i$-th server can read all messages that $\mathsf{P}'$ 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 $\rho$ 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 [37].

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 [22] already obtained MPC with IA by making non-black-box use of the GMW-compiler [29], we claim that the IPS compiler can also be modified so as to yield IA: by replacing two-party OT $\mathcal{F}_{\mathsf{OT}}^2$ with Fully Committed Oblivious Transfer $\mathcal{F}_{\mathsf{FCOT}}^n$, 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 $\rightsquigarrow$ SFE).** *Let $n$ be any number of parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$ and let $f : (\{0,1\}^\lambda)^n \to (\{0,1\}^\lambda)^n$ be an efficiently computable function. There is a protocol $\pi_{\mathsf{SFE}}$ in the $\{\mathcal{F}_{\mathsf{FCOT}}^n\}$-hybrid model that statistically securely UC-realizes $\mathcal{F}_{\mathsf{SFE},f}^n$.*

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

We use the IPS-compiler [37] described above, which compiles two protocols $\Pi$ and $\rho$ into an $\{\mathcal{F}_{\mathsf{OT}}^2\}$-hybrid protocol $\Phi$. 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 $\mathcal{F}_{\mathsf{OT}}^2$-calls with to $\mathcal{F}_{\mathsf{FCOT}}^n$. (3) When a client $\mathsf{P}$ would abort in the original protocol [37] 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,[14] then all honest parties send $(\texttt{conflict}, C')$ to $\mathcal{F}_{\mathsf{CG}}^n$. Or the opened messages indicate no malicious behavior on that server, in which case all honest parties send $(\texttt{conflict}, \mathsf{P})$ to $\mathcal{F}_{\mathsf{CG}}^n$. 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 [37] 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 [37], 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 [37]. 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 $\mathcal{F}_{\mathsf{SFE},f}^n$, and all simulated

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

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 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 $m_c$ associated 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 $\mathsf{P}$ and abort with $(\texttt{abort}, \mathsf{P})$. 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 $(\texttt{abort}, \mathsf{P}')$ where $\mathsf{P}'$ 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 [37], 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 $\Pi$. In the following, we formalize this idea: let $n$ be the number of clients in the outer protocol. In the original paper [37] there are $m \in \Theta(n^2\lambda)$ servers. Each party gets to select $\lambda$ watchlists from each party, such that each party can see all in- and outcoming communication of $\lambda$ 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. □

***Instantiating the Correlated-Randomness model.***

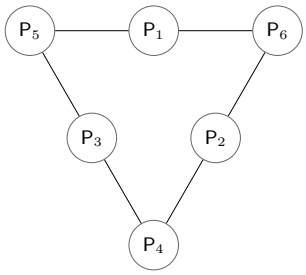
**Lemma 10 (SFE $\rightsquigarrow$ Correlated-Randomness).** *Let $n$ be any number of parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$. For any efficiently samplable distribution $\mathcal{D}$ there is a protocol $\pi_{\mathsf{Corr},\mathcal{D}}$ in the $\{\mathcal{F}^n_{\mathsf{SFE},f}\}$-hybrid model that perfectly UC-realizes $\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}$.*

*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 $\texttt{start}$ 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. □

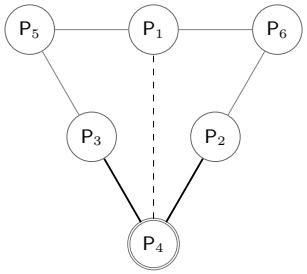
***FCOT in the Correlated-Randomness model.***

**Lemma 11 (Correlated-Randomness $\rightsquigarrow$ FCOT).** *Let $n$ be any number of parties $P = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$. There is a protocol $\pi_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^n_{\mathsf{Corr},\mathcal{D}}\}$-hybrid model that statistically UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$ where $\mathcal{D}$ is the distribution from [36].*

*Proof.* This follows directly from the MPC-completeness of the Correlated-Randomness model in [36]. □




**(a)** Graph is neither 3-settled nor biseparated.




**(b)** Deduction from a given CG. From $\mathsf{P}_1$'s view, $\mathsf{P}_4$ has to be corrupted if $t = 3$.

**Fig. 5:** An example of a graph that is not biseparated but produces a biseparated deduced graph. Thick lines are relevant for the respective property.

## B Relation between biseparation and *t*-settledness

In Section 3 we gave the intuition that $t$-settledness mirrors external explanations and biseparation mirrors internal representations. One might be tempted to assume that $t$-settledness implies biseparation. However, this is—in general—not the case. That is, in a $t$-settled CG the identity of the cheaters is fixed in an information-theoretic sense but it might not be possible to find it efficiently.

An example for this is shown in Fig. 2a. The graph is 1-settled yet not biseparated. Given this graph and the information $t = 2$, all parties can clearly identify $\mathsf{P}_1$ as malicious. If all parties then declare a conflict with $\mathsf{P}_1$, then the CG obviously becomes biseparated with $\{\mathsf{P}_1\}$ and $\{\mathsf{P}_2, ..., \mathsf{P}_5\}$. In this sense the CG can be "completed" by the algorithm DeduceCG.

**Algorithm 1** DeduceCG$(G, t)$

```
1: (P, E) ≔ G
2: changed ≔ 1
3: while changed = 1 do
4:     changed ≔ 0                                   ▷ no change in this iteration yet
5:     for all P ∈ P do
6:         P_red ≔ P \ ({P} ∪ N(P))                  ▷ reduced party set
7:         if P_red is (t − |N(P)|)-settled then
8:             P_× ≔ findIMVC((P_red, E ∩ 2^{P_red}), t − |N(P)|)
9:             for P′ ∈ P_× do
10:                E ≔ E ∪ {P, P′}                   ▷ append inferred conflicts
11:                changed ≔ 1                       ▷ mark change
12:            end for
13:        end if
14:    end for
15: end while
16: return G* ≔ (P, E)                               ▷ deduced Conflict Graph
```

Informally, the algorithm adds new edges whenever a party's view can only be explained by a certain other party being malicious. The algorithm sequentially takes on the role of each party $\mathsf{P} \in P$ in the induced CG $G = (P, E)$. Take for example the graph in Fig. 5a, which is not 3-settled since the vertex covers $\{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3\}$ and $\{\mathsf{P}_4, \mathsf{P}_5, \mathsf{P}_6\}$ do not contain any common party. The algorithm starts by taking the viewpoint of $\mathsf{P}_1$; in this role, any conflict edge of the form $\{\mathsf{P}_1, \mathsf{P}'\}$ implies that $\mathsf{P}'$ is malicious, whereas conflicts between two other parties, e.g. $\{\mathsf{P}_3, \mathsf{P}_5\}$, leaves some uncertainty with respect to the corruptions. Hence, the algorithm computes all explanations conditioned on the fact that the neighbors of $\mathsf{P}_1$ are corrupted, and which are of size at most $t$. Then it adds conflicts between the parties in the explanation and the current party. The algorithm then repeats this step for each party $\mathsf{P} \in P$ until no new conflicts can be found.

More formally, let $\mathrm{N}(\mathsf{P})$ be the neighbors of $\mathsf{P}$ in $G$, that is, all parties $\mathsf{P}'$ for which $\{\mathsf{P}, \mathsf{P}'\} \in E$. The algorithm checks, if the rest of the graph on $P \setminus (\{\mathsf{P}\} \cup \mathrm{N}(\mathsf{P}))$ is $(t - |\mathrm{N}(\mathsf{P})|)$-settled and

appends conflicts between $\mathsf{P}$ and all parties in the settled set of this subgraph.
For the induced Conflict Graph from Fig. 5a with $t = 3$, this would mean that DeduceCG would take on the role of $\mathsf{P}_1$ and check, if the sub-graph on $\{\mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4\}$ is 1-settled. Since the only vertex cover of size 1 contains $\mathsf{P}_4$, DeduceCG adds an edge from $\mathsf{P}_1$ to $\mathsf{P}_4$ in Fig. 5b.
In an information-theoretic sense the DeduceCG algorithm *completes* an induces CG, i.e. a CG returned by $\mathcal{F}_{\mathsf{CG}}^n$. The deduced CG contains all conflicts, even those which might be hard to find via computing the settled set of the induced CG and biseparating it from its complement. This is exactly the reason why DeduceCG is at least as hard as finding the IMVC of the CG.

For our results we require that each edge of the Conflict Graph contains at least one malicious party. While this is obviously the case for edges which are caused by abort of a setup, this may not be obvious for edges which were added by the DeduceCG algorithm. However, we argue that this is the case.

**Lemma 12 (Correctness of** DeduceCG**).** *Let $G = (P, E)$ be a Conflict Graph, where it holds for all edges $\{\mathsf{P}_i, \mathsf{P}_j\} \in E$, that at least one of $\mathsf{P}_i$ and $\mathsf{P}_j$ is corrupt. For all edges $\{\mathsf{P}_i, \mathsf{P}_j\} \in E^*$ of the deduced CG $G^* = (P, E^*) \coloneqq \text{DeduceCG}(G)$, it holds that at least one of $\mathsf{P}_i$ and $\mathsf{P}_j$ is corrupt.*

*Proof.* We prove our claim by contradiction. Let $C \subseteq P$ be the set of malicious parties and let $\{\mathsf{P}_i, \mathsf{P}_j\}$ be a an honest-honest edge in $E^*$. The edge can have only two possible origins. Either it was part of the original Conflict Graph, that is, $\{\mathsf{P}_i, \mathsf{P}_j\} \in E$; this would be a contradiction to our assumption regarding $G = \{P, E\}$, or the edge was added during DeduceCG. From the conditions that have to be fulfilled according to Algorithm 1 in order for an edge to be added, we can infer a lot of information. Without loss of generality, we assume that the edge was added while the view of $\mathsf{P}_i$ was investigated.

Let $P' \coloneqq P \setminus (\{\mathsf{P}_i \cup \mathrm{N}(\mathsf{P}_i)\})$ and let $G' \coloneqq (P', E \cap \binom{P'}{2})$ be the conflict graph that excludes $\mathsf{P}_i$ and its neighbors $\mathrm{N}(\mathsf{P}_i)$. By requirement from Algorithm 1, $G'$ is $t'$-settled for $t' \coloneqq t - |\mathrm{N}(\mathsf{P}_i)|$ and all explanations of $t'$ corrupted parties contain $\mathsf{P}_j$ as corrupted party, i.e. $\mathsf{P}_j \in X(G', t')$. However, the explanation $C' \coloneqq C \setminus (\{\mathsf{P}_i\} \cup \mathrm{N}(\mathsf{P}_i))$ must be a valid explanation for $G'$ since it contains exactly all the malicious parties in $G'$ and $|C'| = |C| - |\mathrm{N}(\mathsf{P}_i)| \leq t'$. By the previous statement it follows that $\mathsf{P}_j \in C' \subseteq C$ which contradicts $\mathsf{P}_j \notin C$. □

***Hardness of completing the CG.*** Now, note that DeduceCG uses the subroutine findIMVC to compute the settled set. If $n \in \mathcal{O}(\ln \lambda)$ or $n - t \in \Theta(1)$, then this problem can trivially be solved in polynomial time in $\lambda$ by intersecting all (at most $\binom{n}{t}$) VCs. However, for a larger number of parties or more corrupted parties, finding the intersection of all VCs of size at most $t$ of arbitrary graphs might be hard. To the best of our knowledge, this exact problem has not been investigate in the literature; we hence study it as the Intersecting Minimal Vertex Cover (IMVC) problem. Here, we show that for arbitrary graphs the IMVC problem is $\mathcal{NP}$-hard by giving a Cook-reduction to the standard MinVC problem. The Decisional Minimal Vertex Cover problem [39] states the hardness of deciding whether a given graph $G$ has a Vertex Cover of size at most $k$. We use the abbreviation MinVC for Minimum VCs and MVC for minimal VCs.

The algorithm hasMinVC describes a Cook-reduction of IMVC problem to the decisional MinVC problem; it requires at most $n^2$ calls to an oracle for the IMVC problem findIMVC to solve the decisional MinVC problem. This yields the following insight: if efficient protocols for Identifiable Abort exists for many parties and many corruptions, then the adversary *cannot* induce arbitrary CGs. The CGs that can be induced by an adversary are limited to the class of CGs from which the cheater's identity is efficiently extractable. Conversely, the class of inducible CGs must be a subset of efficiently extractable CGs.

To analyze why Algorithm 2 yields a valid reduction we have to check its runtime and its correctness. In each outer while-loop, the candidate set $X$ grows by at least one party. Hence, the outer while-loop is executed at most $n$ times. Verifying that a set of nodes corresponds to a VC can be done in time linear in $n$. The oracle findIMVC is called at most once per inner loop-iteration, i.e. at most $n^2$ times. Finally, removing isolated nodes comes down to checking if any nodes without neighbors exist and removing them; this can be done in time $\mathcal{O}(n)$.

Analyzing the correctness is more tricky. Intuitively, the algorithm accumulates a candidate vertex set $X$ that always remains a subset of some MinVC but steadily increases. Our main focus is on the two lines 9 and 11 where the new vertices are chosen that are added to the candidate

**Algorithm 2** hasMinVC($G = (P, E), k$)

```
1:  G' ≔ copy(G)
2:  X ≔ ∅
3:  while !isVC(G, X) do                          ▷ check if is Vertex Cover in O(n)
4:      removeIsolatedNodes(G')                   ▷ O(n)
5:      X' ≔ ∅
6:      t ≔ 0
7:      while X' = ∅ do                           ▷ at most |G'| ≤ n iterations
8:          if t < |G'| then
9:              X' ≔ findIMVC(G', t)
10:         else
11:             X' ≔ {P_next}                     ▷ P_next ∈ G' lexicographically smallest
12:         end if
13:         t ≔ t + 1
14:     end while
15:     X ≔ X ∪ X'                                ▷ |X| strictly increases until it becomes VC
16:     G' ≔ G' \ X'                              ▷ remove nodes X' and incident edges
17:     if |X| > k then
18:         return 0
19:     end if
20: end while
21: return 1
```

set $X$. In both cases we have to ensure that the nodes of $X'$ are in a common MinVC with the already chosen $X$. If line 9 gets executed, then $X' \coloneqq \text{findIMVC}(G', t)$ is chosen as vertices that are in all remaining MVC. Hence, adding these vertices to $X$ still yields a subset of some MinVC of $G$. If line 11 gets executed, no IMVC exists in $G'$. The reduction adds the lexicographically first node (this could as well be a random node without any loss of correctness). The reason that any vertex can be added to $X$ while maintaining a subset of a MinVC is that each vertex is contained in at least one MinVC of $G'$. If there was one vertex without a corresponding MinVC, then all its neighbors—whose existence is guaranteed by removeIsolatedNodes—must be in all MVCs, in contradiction to the case condition.
The loop invariant that $X$ is a subset of some MinVC of $G$ is maintained in both cases. Finally, we get either that $X$ becomes a VC, thus the MinVC answer is true, or the size of $X$ exceeds $k$, then the size of each MinVC of $G$ must also exceed $k$, the answer is false.

*Remark 3.* Note that our SFE-expansion[15] in Section 3 doesn't suffer from the logarithmic limitation of $n$ because we actually don't invoke the DeduceCG algorithm.

[15] More precisely any constant number of recursions.