# Constructing Secure Multi-Party Computation with Identifiable Abort

Nicholas-Philip 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.** Statistically secure Multi-Party Computation (MPC) protocols based on *two-party* primitives like Oblivious Transfer [Kil88; IPS08] have one severe drawback: the adversary can abort the protocol without repercussions if the majority of all parties are malicious. To evade impossibility of fairness [Cle86], the notion of Identifiable Abort (IA) was introduced in [IOZ14]; here cheaters are exposed upon abort.
Given a broadcast, we tightly link the unanimous identifiability of any protocol to verifiable graph-theoretical properties using our new Conflict Graph (CG)-technique. As such, we formalize a necessary and sufficient CG property that any protocol must fulfill in order to guarantee security with IA. We conjecture that for certain instances, Identifiable Abort is actually an NP-hard problem.
Furthermore, we leverage our Conflict Graph in a concrete construction to give the first upper bound of the minimal setup size, in the sense of [FGM$^+$01], for $n$-party MPC in the dishonest majority setting. That is, in the IA-setting we show that $n$-party statistically Secure Function Evaluation (SFE) can be composed from $(n-1)$-party SFE and broadcast if the maximal number of corruptions is $t \leq (n-3)$. Additionally, if the number of parties is sufficiently small, then our upper bound can be transitively expanded: for $t \coloneqq (n-k-3)$ corruptions, we can construct $n$-party SFE form $(n-k-1)$-party SFE.

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

## 1 Introduction

Secure Multi-Party Computation (MPC) has been subject to extensive studies since the 1980's. The requirements for general MPC have been studied extensively in the literature for a variety of settings. Most notably, MPC protocols have been constructed from abstract assumptions such as Oblivious Transfer (OT) instead of concrete computational assumptions [Kil88; IPS08]. Since the introduction of the real-ideal-paradigm [GMW87] MPC has been based on so-called hybrid functionalities, such as a Common Reference String (CRS), which is commonly used e.g. in the Universal Composability (UC)-framework of [Can01]. These hybrid functionalities exhibit the distinct advantage that they can be based on a variety of computational assumptions such as Discrete Logarithm, Learning with Errors or Integer Factorization, or even physical assumptions such as noisy channels [GIS$^+$10; CK88; Cré97].

The systematic study of the minimal size—or *minimal complete cardinality*—of an MPC-setup has been initialized by [FGM$^+$01]. In the honest-majority case there have been several results, among others, [GMW87; BGW88; FGM$^+$01] without Broadcast, and [RB89; Bea90] with a Broadcast. More than 30 years ago [RB89; Bea90] showed that, for an honest majority, pairwise secure channels are sufficient in conjunction with a Broadcast (even for Guaranteed Output Delivery). In the terminology of [FGM$^+$01] the minimal complete cardinality is 2. Against a dishonest majority, the minimal complete cardinality is also 2 [Kil88; IPS08] but only for protocols with Anonymous Abort. This is consistent with the impossibility of fairness in the dishonest majority setting [Cle86].

However, security with Anonymous Abort allows an adversary to effectively perform a Denial-of-Service-attack on the protocol without being detected. As a more promising alternative, the notion of Identifiable Abort (IA) was introduced in [IOZ14]. Here at least one malicious party is identified by all honest parties upon abort. Intuitively, this allows the other parties to exclude the identified culprit in the next protocol run. [1] [IOS12] showed that any functionality of cardinality 2 is insufficient for general MPC with cheater identification (IA) without Broadcast. Yet the first official formalization of Identifiable Abort [IOZ14] already provided a universal setup of size $n$ called *Correlated Randomness*-model.

To enhance the understanding of IA we investigate requirements for protocols to unanimously identify a cheater assuming a Broadcast is available to each party. Using our Conflict Graph (CG)-technique we initiate the study on the dishonest majority setting by linking IA to graph-theoretical properties of the CG. We give the first upper bound for the minimal complete cardinality with Broadcast in the UC-framework and dishonest-majority setting with IA.

While most research in the area of Identifiable Abort has been directed towards the efficiency of concrete MPC protocols (such as [DPS$^+$12; SF16; BOS16]), our construction focuses on a feasibility result showcasing the power of our CG-technique. Yet we believe that the CG will also prove useful in the construction of efficient protocols with IA.

*Outline* In Section 1.1 we summarize our main contributions. After specifying the considered security notions in Section 1.2 we give a high-level overview of our results in Section 1.3. In 2, we define the used notations and definitions. This is followed by the main part of our work; we provide more detailed descriptions and detailed proofs of our Conflict Graph in Section 3. We provide several constructions in Sections 4 to 7. Finally, we conclude with a summary and an outlook in Section 8.

[1] The protocol has to be designed such that the aborted functionality does not leak sensitive information that could be used by the adversary in future protocol runs.

## 1.1 Contribution

Our three main contributions are:

***Conflict Graph.*** We introduce the *Conflict Graph (CG)* and tightly link the Identifiable Abort property of any MPC-protocol to verifiable graph-theoretical properties of CG. While the Conflict Graph is of theoretical interest on its own—we link Identifiable Abort to a potentially NP-hard problem of the CG, which could provide further insight to Identifiable Abort upon further investigation—we also use it to construct a new MPC-protocol.

Technically, the Conflict Graph is a graph $G = ([P], E)$, where $[P]$ is the set of parties. $G$ has a vertex for each party $\mathsf{P} \in [P]$. An edge $e = \left\{\mathsf{P}, \mathsf{P}'\right\} \in E$ stands for a publicly declared conflict between the two parties $\mathsf{P}$ and $\mathsf{P}'$; we require that conflicts between honest parties never arise, this is also a guarantee that either $\mathsf{P}$ or $\mathsf{P}'$ is corrupted.

On an abstract level, once sufficiently many conflicts have been declared the honest parties can leverage this information to unanimously identify a set of cheaters. In this work, we provide necessary and sufficient conditions for an IA which facilitates our MPC-construction.

***New Oblivious Transfer variant.*** We reformulate Crepeau's *Committed* OT [Cré90; CvT95] in the multi-party setting. We call our variant Fully Committed Oblivious Transfer (FCOT) and show its usefullness in the setting of Identifiable Abort. In an FCOT all 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 unveil them at a later point to all other parties; even after the receiver obtained only one message, the sender can unveil both $m_0$ *and* $m_1$, and the receiver can unveil the choice bit $c$, such that no party can lie about their actual input.

We also prove equivalence of Secure Function Evaluation and Fully Committed Oblivious Transfer in the setting of Identifiable Abort. That is, we show how to instantiate $n$-party SFE in the FCOT-hybrid-model and how to realize $n$-party FCOT using only a SFE hybrid functionality.

***Expanding SFE with IA.*** Finally, we provide an expansion from $(n - 1)$-party MPC and $n$-party Broadcast to $n$-party MPC. This implies an upper bound for the minimal complete cardinality for $n$-party Secure Function Evaluation (SFE) in the style of [FGM$^+$01], assuming that at most $(n-3)$ parties are malicious. If the number of parties is sufficiently small, we can extend our result by induction, yielding a better bound for the dishonest majority setting (compare Fig. 1).

More precisely, we give a protocol that expands FCOT from cardinality $(n - 1)$ to $n$. Since FCOT is equally powerful as SFE this implies an expansion of SFE from $(n - 1)$ to $n$. As an intermediate result we expand a Commitment from size $(n - 1)$ to $n$ and then use the $n$-party Commitment to ensure consistency across all instances of the $(n - 1)$-party FCOT.

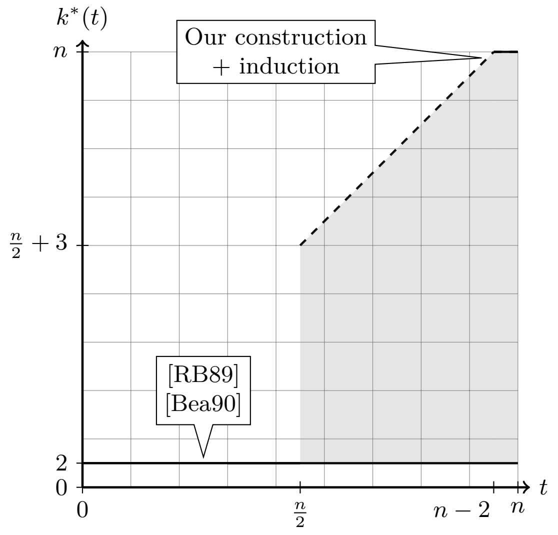


**Fig. 1:** Bounds of the minimal complete cardinality $k^*(n,t)$ with IA vs. *maximal* number of malicious parties $t$ given broadcast. The grey area represents the possible region of $k^*(n,t)$. The dashed lines indicate our bounds (linear induction for $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$).

### 1.2 Setting

*All* of our constructions enjoy **statistical security**, no computational assumptions are made. We only assume the existence of hybrid functionalities. This leaves the means of the realization of these hybrid functionalities up to the user, e.g. via physical means such as trusted hardware [GIS+10; SSW10] or noisy channels [CK88; Cré97], or again from computational assumptions with better efficiency [Bon98; Reg05]. We don't explicitly assume additional pairwise secure channels, since they can be emulated by hybrid functionalities 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 $\mathcal{I}^n_{\mathsf{BC}}$. This broadcast is mainly used for our realization of the Conflict Graph, for more details see Section 5.

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. It only learns that input was provided.

Finally, for technical reasons, our construction requires a limit on the number of parties to $n \in \mathcal{O}(\ln\lambda) \lor n - t \in \mathcal{O}(1)$ to efficiently identify malicious parties. We do not currently know whether this restriction is inherent. Though, note that the strongest adversarial case $t+1 = n \in \mathrm{poly}(\lambda)$ is covered in our construction. However, we want to emphasize that while this seems like a weakness of the

Conflict Graph, it is actually a general lower bound on unanimous Identifiability: we show in Theorem 7 that the graph properties that seem to necessitate the limitation of $n$ are indeed necessary for an unanimous identification. For more details on this see Section 3.

*The UC Framework* We perform our analysis in the Universal Composability framework [Can00; Can01], which is a strong version of simulation-based security [GMW87]. 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 parties inputs. The behavior of this party is specified by a *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. The transcript includes the output of all parties and the respective adversary. Indistinguishability 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-model is strictly stronger than the standalone model; without a trusted setup, Commitments are not possible in the UC-model [CF01] while they can be constructed from computational assumptions in the standalone model. Constructions in the UC-model also hold in the standalone model and, conversely, impossibilities in the standalone model extend to the UC-model.

We assume a *synchronous* communication network, as our Conflict Graph requires that any conflict announced by a party $\mathsf{P}$ will eventually be received by all other parties. In an asynchronous model, the adversary could drop all messages [CM89; BCG93], 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 unveiling the identity of at least one malicious party. We adapt the view from the 2020 version[2] of [Can00], which describes how synchronous communication can be achieved by using the functionality $\mathcal{F}_{\mathsf{SYN}}$. For the sake of simplicity, however, we ignore the details of $\mathcal{F}_{\mathsf{SYN}}$ in our analysis and just assume a synchronous communication structure.

*Identifiable Abort* Constructing protocols requires definitions regarding the abort properties. Intuitively, the most desirable property is *guaranteed output*, where an abort is impossible. Unfortunately, fairness and thus guaranteed output is impossible with a dishonest majority [Cle86]. On the other extreme, the much weaker notion of Anonymous Abort leaves the adversary capable of stopping any computation without repercussions and thus is an undesirable property for many real-world scenarios.

We work in the setting of IA [IOZ14], where abort is possible, but only by revealing the identity of (at least) one malicious party to all participants. In this

[2] Version 20200212:021048

setting, all honest parties eventually expel all malicious parties, if the protocol is aborted too often. Thereby it suffices that, during an unsuccessful protocol run, all honest parties agree on at least one disruptor. Then the adversary can abort the protocol at most $(n-1)$ times, before all malicious parties are excluded or the protocol succeeds.

We use the following notation to clarify our Identifiable Abort property[3]:

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

**Definition 1 (Identifiable Abort).** *Let $\mathcal{I}^n$ be an ideal $n$-party functionality with parties $[P]$ and malicious subset $C \subseteq [P]$. $\mathcal{I}^n$ has **Multi-Identifiable Abort**, iff all (honest) parties yield output $\left(\texttt{abort}, C'\right)$ when the adversary sends $\left(\texttt{abort}, C'\right)$ to $\mathcal{I}^n$. If $C' \nsubseteq C$, the message is ignored. $\mathcal{I}^n$ has **Uni-Identifiable Abort**, iff $\mathcal{I}^n$ has* Multi-IA *and $|C'| = 1$.*

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 one protocol run, which it can leverage during the next execution. The honest parties then neither learn their output, nor have a precise estimate on how sensitive the data obtained by the adversary is. By using secret-sharing schemes, we are still able to compose our functionality *Fully Committed Oblivious Transfer (FCOT).*

### 1.3 Overview

*Conflict Graph* We introduce and analyze the Conflict Graph $G$ and prove a tight linkage between verifiable graph properties and the identification of disruptors in the field of Identifiable Abort. The Conflict Graph maintains the global view of conflicts that arise between parties during a protocol execution.

Let $\pi$ be an $n$-party protocol that uses the Conflict Graph. The CG of $\pi$ is an undirected, simple graph $G = ([P], E)$ where $[P] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ is the set of parties of $\pi$. Intuitively, during the protocol execution a party $\mathsf{P}$ that notices any misbehavior of another party $\mathsf{P}'$ publicly announces that it is in conflict with $\mathsf{P}'$. While honest parties only issue conflicts with malicious parties, malicious parties can issue conflicts with any party. It thus holds for each conflict edge that at least one of the two parties is malicious.

On a high level, we either want a protocol execution to successfully terminate, or all honest parties to settle on the same set of corrupted parties. Once sufficiently many conflicts are issued honest parties can leverage the Conflict Graph to unanimously identify parties that actively deviate from the protocol, which we call *disruptors*. In particular, any party that aborts a subfunctionality (hybrid functionality) is considered a disruptor. Note that the precise condition of a conflict declaration based on protocol deviations is highly protocol-dependent;

[3] Note that the original work [IOZ14] uses the notation $\mathcal{F}_{\perp}^{\mathsf{ID}}$.

we defer the specifics of our constructions to Section 7. The formal description of the Conflict Graph functionality looks as follows:

Functionality $\mathcal{I}^n_{\mathsf{CG}}$

$\mathcal{I}^n_{\mathsf{CG}}$ 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 $E \coloneqq \emptyset$.
- When receiving a message $(\mathtt{conflict}, \mathsf{P}_i)$ from $\mathsf{P}_j$, append the new conflict edge $\{\mathsf{P}_i, \mathsf{P}_j\}$ to the set of conflict edges $E$ and send $(\mathtt{conflict}, \mathsf{P}_j, \mathsf{P}_i)$ to the adversary.
- When receiving a message $(\mathtt{query})$ from $\mathsf{P}_i$, deduce the current Conflict Graph $G^* \coloneqq \text{DeduceCG}(([P], E), t)$ and output $G^*$ to $\mathsf{P}_i$.

When receiving $\big(\mathtt{abort}, C'\big)$ from $\mathcal{S}$ with $C' \subseteq C$, then $\mathcal{I}^n_{\mathsf{CG}}$ outputs $\big(\mathtt{abort}, C'\big)$ to all parties, and then terminates.

The functionality internally maintains the graph. Regardless which party queries the current Conflict Graph, $\mathcal{I}^n_{\mathsf{CG}}$ returns the same graph. This implies a *consistent* view to all (honest) parties for any protocol $\pi^n_{\mathsf{CG}}$ that realizes $\mathcal{I}^n_{\mathsf{CG}}$.

Keeping in mind that each conflict edge contains at least one corrupted party, we call any subset of parties which could have caused the structure of Conflict Graph an *explanation* of $G$. Intuitively, an explanation of a Conflict Graph $G$ comes down to a *vertex cover* of $G$. We separate between *internal* and *external* explanations. External explanations allow even non-participants to easily identify a set of disruptors, given only the Conflict Graph and the maximum number $t$ of corrupted parties. Internal explanations are limited to explanations of the graph for a given party $\mathsf{P}$, which knows that itself is behaving honestly.

When queried for the current Conflict Graph, $\mathcal{I}^n_{\mathsf{CG}}$ invokes the algorithm called DeduceCG from Algorithm 1. We call the graph that arises only from declared conflicts the *induced* CG $G$ while we refer to the output of the $\mathcal{I}^n_{\mathsf{CG}}$ functionality as the *deduced* CG $G^*$. Before we go into detail about the DeduceCG-algorithm, we have to introduce some properties of the CG and the relation between them. For now it suffices to think of DeduceCG as a mechanism for honest parties to *complete* the CG with implicit conflicts that have not explicitly been declared.

We now show that the properties required for identifying disruptors can be brought down to properties of the Conflict Graph. The simple idea is that, once sufficiently many conflicts have arisen, at least one common party must be contained in *all* explanations.

**Definition 2 ($t$-settledness).** *Let $n$ be the number of parties $[P]$ of which at most $0 \leq t < n$ are malicious. Let $G^* = ([P], E)$ be the Conflict Graph of a protocol $\pi$. Let $M(G, t)$ be the set of all Minimal Vertex Covers (MVCs) of $G^*$ with size $t$ or less, and let $X(G^*, t)$ be the intersection of all of these MVCs, that* *is, the set of parties which are present in all MVCs of size* $\leq t$*. We call* $X(G^*, t)$ *the* settled set *of* $G^*$*. We call* $G^*$ $t$-settled*, iff* $X(G^*, t) \neq \emptyset$*.*

The definition already showcases the equivalence between $t$-settledness of a Conflict Graph and its *external explanation*—each Minimal Vertex Cover of size $\leq t$ is a valid explanation of corrupted parties that *could* have caused this graph, and if each possible explanation implies that a party $\mathsf{P}$ is corrupted, then even non-participants can be convinced that party $\mathsf{P}$ is a disruptor.

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

```
1:  $([P], E) \coloneqq G$
2:  $changed \coloneqq 1$
3:  **while** $changed = 1$ **do**
4:      $changed \coloneqq 0$                                   ▷ no change in this iteration yet
5:      **for all** $\mathsf{P} \in [P]$ **do**
6:          $[\tilde{P}] \coloneqq [P] \setminus (\{\mathsf{P}\} \cup \mathrm{N}(\mathsf{P}))$                         ▷ reduced party set
7:          **if** $[\tilde{P}]$ is $(t - |\mathrm{N}(\mathsf{P})|)$-settled **then**
8:              $[P]_\times \coloneqq X(([\tilde{P}], E \cap 2^{[\tilde{P}]}), t - |\mathrm{N}(\mathsf{P})|)$     ▷ settled set from Definition 2
9:              **for** $\mathsf{P}' \in [P]_\times$ **do**
10:                 $E \coloneqq E \cup \{\mathsf{P}, \mathsf{P}'\}$                    ▷ append inferred conflicts
11:                 $changed \coloneqq 1$                                  ▷ mark change
12:             **end for**
13:         **end if**
14:     **end for**
15: **end while**
16: **return** $G^* \coloneqq ([P], E)$                              ▷ deduced Conflict Graph
```

With the definition of $t$-settledness in mind, we can now analyse the algorithm DeduceCG. Let's start with an example Fig. 2c, we can see a graph 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. Note that this graph can only ever appear as an induced CG, as we will see shortly. We use the notation of Algorithm 1. The algorithm sequentially takes on the role of each party $\mathsf{P} \in [P]$ in the induced CG $G = ([P], E)$. From the viewpoint of $\mathsf{P}$, any conflict edge of the form $\{\mathsf{P}, \mathsf{P}'\}$ implies that $\mathsf{P}'$ is a disruptor, whereas conflicts between two other parties $\{\mathsf{P}'', \mathsf{P}'''\}$ leave some uncertainty regarding which of the two might be an actual disruptor. Hence, $\mathsf{P}$ checks for those explanations, which contain all of its declared conflicts, and which are of size $\leq t$. 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 for each party $\mathsf{P}$, if the rest of the graph on $\widetilde{[P]} \coloneqq [P] \setminus (\{\mathsf{P}\} \cup \mathrm{N}(\mathsf{P}))$ is $(t - |\mathrm{N}(\mathsf{P})|)$-settled and appends conflicts between $\mathsf{P}$ and all parties in the settled set $[P]_\times$ of the subgraph on $\widetilde{[P]}$.

For the induced Conflict Graph from Fig. 2c, 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. 2f.

The functionality $\mathcal{I}^n_{\mathsf{CG}}$ can be realized using only a Broadcast-functionality $\mathcal{I}^n_{\mathsf{BC}}$ (see Lemma 13). The protocol lets every party maintain an induced CG from the broadcasted conflicts and each party can then produce the deduced CG *locally* by invoking DeduceCG. In this sense the functionality $\mathcal{I}^n_{\mathsf{CG}}$ is merely a convenient way of handling the Conflict Graph. We emphasize that it is *not* an additional assumption if a broadcast is given.

Now we have seen how the $t$-settledness affects the Conflict Graph, i.e. if external observers can identify disruptors. However, for Identifiable Abort, it is not necessary for participants to convince external parties, but only participants must be able to identify disruptors. We therefore introduce a different graph property which reflects valid explanations for participants only:

**Definition 3 (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 $[P]$.*

Thus, when a Conflict Graph is biseparated, it can be partitioned into two partitions. Each party agrees with its own partition that all parties from the other partition are malicious. Consequently, all honest parties must be in one partition (possibly among some malicious parties).

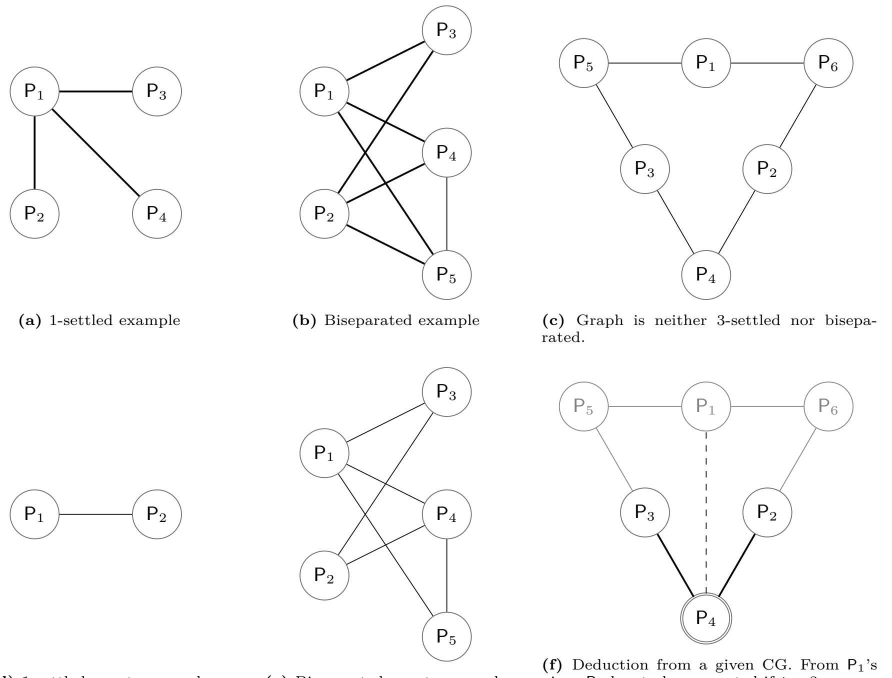
While biseparation of a graph $G$ is decidable in linear time in $n$ by a breath-first-search to check whether the complement graph is connected, deciding on $t$-settledness is much harder.

The DeduceCG algorithm requires the computation of the settled set $X(G, t)$. To our knowledge the best known algorithm for this requires computation of the intersection of all vertex covers of size $t$. This way $\binom{n}{t} = \binom{n}{n-t}$ subsets of size $t$ have to be tested if they are vertex covers, thus this trivial algorithm certainly runs in $\mathrm{DTIME}\big(\binom{n}{t}n\big)$. For efficiency this must be polynomial in $\lambda$. We describe two special efficient cases: First, for arbitrary $0 \leq t < n$ we must limit $n \in \mathcal{O}(\ln \lambda)$. Second, for $t \in \mathcal{O}(1)$ or $n - t \in \mathcal{O}(1)$ we can support any polynomial $n \in \mathrm{poly}(\lambda)$. The case of $t \in \mathcal{O}(1)$ is uninteresting since this implies an honest-majority. Hence we obtain the condition $n \in \mathcal{O}(\ln \lambda) \vee n - t \in \mathcal{O}(1)$ which is used throughout our constructions.

Surprisingly, it seems easier to identify cheaters for $t = (n - c)$ for a constant $c$ than for $t = n/2$; if we allow a more powerful adversary, it seems to become easier to mitigate its attacks. For further research we pose the question: given $n$ and $t$, what is the time complexity of computing $X(G, t)$ in general?

We generally assume that protocols use the Conflict Graph *correctly*, meaning that honest parties immediately report a conflict with disruptors. While this seems like a restriction at first, we show that it is not:

**Lemma 1 (Informal version of Lemma 10).** *Let $n$ be the number of parties $[P]$ of which at most $0 \leq t < n$ are malicious. Let $\pi$ be a protocol that securely UC-realizes a functionality $\mathcal{I}^n$ in the some $M$-hybrid model without $\mathcal{I}^n_{\mathsf{CG}}$. Denote*



**(a)** 1-settled example **(b)** Biseparated example **(c)** Graph is neither 3-settled nor biseparated.

**(d)** 1-settled counterexample **(e)** Biseparated counterexample **(f)** Deduction from a given CG. From $\mathsf{P}_1$'s view, $\mathsf{P}_4$ has to be corrupted if $t = 3$.

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

*by* $\pi'$ *the* $M \cup \{\mathcal{I}^n_{\mathsf{CG}}\}$*-hybrid protocol that is identical to* $\pi$ *with the addition that honest parties use* $\mathcal{I}^n_{\mathsf{CG}}$ correctly*.*

*Then* $\pi'$ *also securely UC-realizes* $\mathcal{I}^n$*, i.e.* $\pi' \geq \mathcal{I}$*.*

This lemma allows us to quantify only over protocols that use the Conflict Graph *correctly*, while still obtaining meaningful results for all protocols with Identifiable Abort. Furthermore, it implies that no adversary gains any advantage when using the Conflict Graph *correctly*.

In Section 3 we elaborate on the relation between $t$-settledness and biseparation. For now we just remark that $t$-settledness of the induced CG implies biseparation of the corresponding *deduced* CG: if $G$ is $t$-settled then DeduceCG will bring the settled set $X(G,t)$ in conflict with all other parties, thereby biseparating the CG $G^*$.

*Remark 1 (*DeduceCG *biseparates t-settled graphs).* Let $G$ be a $t$-settled induced Conflict Graph. The deduced CG $G^* \coloneqq \text{DeduceCG}(G,t)$ is biseparated.

A few example graphs are shown in Fig. 2. Figure 2a shows a graph structure where the only vertex cover of size 1 is $\{\mathsf{P}_1\}$, thus $\mathsf{P}_1$ is in all external explanations, making the graph 1-settled. Note that this graph is also biseparated. The graph from Fig. 2d on the other hand is not 1-settled; both $\{\mathsf{P}_1\}$ and $\{\mathsf{P}_2\}$ are valid Vertex Covers. However, biseparation trivially holds. Another example for biseparation is the Conflict Graph from Fig. 2b. This graph can be split up into two partitions $S_0$ and $S_1$, where $S_0 \coloneqq \{\mathsf{P}_1, \mathsf{P}_2\}$ and $S_1 \coloneqq \{\mathsf{P}_3, \mathsf{P}_4, \mathsf{P}_5\}$. This implies that all members of $S_1$ are convinced that $\mathsf{P}_1$ and $\mathsf{P}_2$ are corrupted, thus allowing them to continue without them. However, if we remove the conflict between $\mathsf{P}_2$ and $\mathsf{P}_5$ (see Fig. 2e), the graph is no longer biseparated, thus $\mathsf{P}_5$ would not agree to throw out $\mathsf{P}_2$.

Now we state our main result: the equivalence of biseparation and Identifiable Abort.

**Theorem 2 (Informal version of Theorem 7).** *Let $\pi$ be a protocol that securely UC-realizes a functionality $\mathcal{I}^n$ with Identifiable Abort in an $M \cup \{\mathcal{I}^n_{\mathsf{CG}}\}$-hybrid model. Upon abort, the deduced Conflict Graph $G^*$ of $\pi$ must be biseparated.*

The proof of this theorem can be found in Theorem 7.

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

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

*SFE-Completeness of FCOT* We reformulate Crepeau's Committed OT [Cré90; CvT95]) for $n$-parties as Fully Committed Oblivious Transfer (FCOT) and formally prove its equivalence to Secure Function Evaluation in the setting of Identifiable Abort. This has the advantage that we can perform our analysis of SFE-expansion by only expanding FCOT. We thus show in two lemmas that one implies the other.

**Lemma 2 (Informal version of Lemma 11).** *For every $n$, there is a protocol $\pi^n_{\mathsf{FCOT}}$ in the $\{\mathcal{I}^n_{\mathsf{SFE}}\}$-hybrid model that securely realizes $\mathcal{I}^n_{\mathsf{FCOT}}$.*

This implies that $n$ parties can use SFE in order to obtain a Fully Committed Oblivious Transfer. We only provide a brief description of the protocol here; the full protocol alongside with a security proof is presented in Lemma 12. The protocol uses four calls to $\mathcal{I}^n_{\mathsf{SFE}}$, which are used for *distributing* and *collecting* secret shares in such a way, that parties obtain *enough* information to stop a dishonest sender or receiver from opening up different values, but *insufficient* data to reconstruct the messages.

Prior to the first call, the sender $\mathsf{S}$ computes $n$ additive secret shares $\mu^b_{j,i}$ of the two messages $m_b$ and the receiver $\mathsf{R}$ computes $n$ additive secret shares $\gamma_{j,i}$ of the choice bit $c$. Every party, including sender and receiver, also choose a random number $\nu$. This determines the theoretical input of each party; every party would want to input $\nu$, $\mathsf{S}$ additionally wants to input all shares $\mu^b_{j,i}$, and $\mathsf{R}$ additionally wants to input the shares $\gamma_{j,i}$. However, each party creates $r$ shares of the input, which are then used as *actual* input for the first function evaluation with $\mathcal{I}^n_{\mathsf{SFE}}$. It outputs the message $m_c$ to $\mathsf{R}$; each party additionally obtains the $\nu$ input of $\mu^0$, $\mu^1$ and $\gamma$. The next two instances of $\mathcal{I}^n_{\mathsf{SFE}}$ are used for the unveiling of the sender's message $m_0$ and $m_1$, respectively, and the last instance of $\mathcal{I}^n_{\mathsf{SFE}}$ opens all shares of the receiver's choice bit.

**Lemma 3 (Informal version of Lemma 12).** *For every $n$, there is a protocol $\pi^n_{\mathsf{SFE}}$ in the $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$-hybrid model that securely realizes $\mathcal{I}^n_{\mathsf{SFE}}$.*

Constructing SFE from FCOT is similar to the construction from Ishai, Prabhakaran, and Sahai [IPS08]; in fact, in the framework of Identifiable Abort, we show in Lemma 11 that it suffices to use their exact protocol, but replacing each invocation of Oblivious Transfer with an invocation of Fully Committed Oblivious Transfer. Upon abort critical FCOTs will be unveiled, thus exposing the disruptors.

*Global Commitment from Fully Committed Oblivious Transfer* We prove that $n$-Party Global Commitment can be constructed from $n$-Party Fully Committed Oblivious Transfer without losing the IA property:

**Lemma 4 (Informal version of Lemma 14).** *For every $n$, there is a protocol $\pi^n_{\mathsf{COM}}$ that securely realizes $\mathcal{I}^n_{\mathsf{COM}}$ in the $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$-hybrid model.*

The protocol uses the *unveil* phase of Fully Committed Oblivious Transfer: For the *commit*-phase, the committer $\mathsf{C}$ acts as *Receiver* in the single instance of $\mathcal{I}^n_{\mathsf{FCOT}}$ and inputs the to-be-committed bit $m$ as choice bit, the first actual receiver $\mathsf{R}_1$ of $\mathcal{I}^n_{\mathsf{COM}}$ acts as Sender in $\mathcal{I}^n_{\mathsf{FCOT}}$ and sends twice the same randomly chosen message, the remaining receivers $(\mathsf{R}_2, \ldots, \mathsf{R}_n)$ participate as *witnesses*. For the *unveil*-phase, $\mathsf{C}$ sends input (`unveil choice`) to $\mathcal{I}^n_{\mathsf{FCOT}}$. The security of this protocol follows from the security guarantees of $\mathcal{I}^n_{\mathsf{FCOT}}$.

*SFE expansion* Recall that the biseparation allows the protocol to abort consistently. We introduce three relevant lemmas, limiting the amount of subfunctionalities of size $(n-1)$ in an $n$-party protocol before the graph becomes biseparated, effectively stating an upper bound on how many subfunctionalities an adversary can abort. We use this guarantee to construct an extension from Commitment and Fully Committed Oblivious Transfer (thus effectively Secure Function Evaluation) from size $(n-1)$ to size $n$.

Note that, for simplicity only, we require protocols to use the CG *correctly* as described in Lemma 10. The most general version of our lemma regarding subfunctionality abort is the following:

**Lemma 5 (Informal version of Lemma 15).** *Let $n$ be the number of parties $[P]$ of which at most $0 \le t < n$ are malicious. Let $\pi$ be an $\left\{\mathcal{I}^{n-1}, \mathcal{I}^n_{\mathsf{CG}}\right\}$-hybrid protocol that uses $\mathcal{I}^n_{\mathsf{CG}}$ "correctly". If the adversary $\mathcal{A}$ aborts more than $t$ instances of $\mathcal{I}^{n-1}$, then the Conflict Graph from $\mathcal{I}^n_{\mathsf{CG}}$ is biseparated.*

We refer to all subfunctionalities that use the same set of parties as the same instance. For functionalities of cardinality $n-1$ there are exactly $n$ instances, one for each party that is excluded. Multiple executions of functionalities on the same set of parties are considered to belong to the same instance.

A formal proof of this lemma is deferred to Lemma 15. Here we only sketch the proof. We denote the instance, where all parties but $\mathsf{P}_i$ participate, as $\mathcal{I}_{\mathsf{P}_i}^{n-1}$. Note that for any $i \in [n]$, after $\mathcal{I}_{\mathsf{P}_i}^{n-1}$ has been aborted, the Conflict Graph $G^*$ contains a biseparated subgraph over all parties except for $\mathsf{P}_i$. We can investigate the complement graph $\overline{G}$. After an abort of $\mathcal{I}_{\mathsf{P}_i}^{n-1}$, the Complement Graph of the participants without $\mathsf{P}_i$ is split into two partitions, which are only connected via $\mathsf{P}_i$ in $\overline{G}$. We abuse notation and write $\overline{G} \setminus \mathsf{P}_i$ for the Complement Graph where $\mathsf{P}_i$ is removed. Say $\mathcal{I}_{\mathsf{P}_i}^{n-1}$ was aborted by $\mathsf{P}_j$. If the same party $\mathsf{P}_j$ aborts a second subfunctionality in which $\mathsf{P}_i$ is participating, its partition becomes completely disconnected in the Complement Graph, as $\mathsf{P}_i$ would also declare conflict with $\mathsf{P}_j$. Thus, the adversary cannot re-use a party to abort a different subfunctionality. Since there are only at most $t$ corrupted parties, we have an upper bound on the number of subfunctionalities the adversary can abort before the Conflict Graph becomes biseparated.

We can furthermore specify this result with respect to certain types of adversaries. Against adversaries who can corrupt any subset of parties of size $t \leq (n-3)$, the lemma tightens to:

**Lemma 6 (Informal version of Lemma 16).** *Let $t \leq (n-3)$. If $t$ or more subfunctionalities of cardinality $(n-1)$ are aborted, then the Conflict Graph is biseparated.*

By requirement, there are at least three honest parties. After $t$ aborts, any party $\mathsf{P}_j$ which aborted $\mathcal{I}_{\mathsf{P}_i}^{n-1}$ can only have two neighbors in the Complement Graph: party $\mathsf{P}_i$, which did not witness the abort, and some other party $\mathsf{P}_k$ who aborted $\mathcal{I}_j^{n-1}$. This is true for all of the $t$ parties who aborted a subfunctionality. Since no conflicts ever arise between honest parties, they have edges in $\overline{G}$ with all other honest parties. With $t \leq (n-3)$, each honest party has at least two honest neighbors in $\overline{G}$. Additionally, each honest party $\mathsf{P}_l$ has an edge with the party that aborted $\mathcal{I}_l^{n-1}$. Thus, after $t$ aborts, only honest parties have three neighbors in $\overline{G}$, allowing them to identify disruptors.

Our next lemma provides a less strict guarantee against a stronger adversary, which can corrupt up to $(n-1)$ parties. Here, Lemma 15 yields:

**Lemma 7 (Informal version of Lemma 17).** *Let $t < n$. If more than $(n-2)$ subfunctionalities of cardinality $(n-1)$ are aborted, then the Conflict Graph is biseparated.*

Let $\mathsf{P}_i$ be an honest party. After an abort of $\mathcal{I}_{\mathsf{P}_i}^{n-1}$, the remaining Conflict Graph $G^*$ is biseparated. This means that the remaining parties can be separated into two partitions $S_0$ and $S_1$, which are disconnected in the Complement Graph.

When an other subfunctionality $\mathcal{I}_j^{n-1}$ that omits party $\mathsf{P}_j$ is aborted, it can only be by a subset of parties from the same partition as $\mathsf{P}_j$, as otherwise, the graph becomes biseparated. This means that $\mathsf{P}_j$ can no longer be used for future aborts, without causing a biseparation. Thus, for each abort, the set of possible disruptors decreases by one, since the omitted party cannot be used as future disruptor. Hence, in order to avoid a biseparated Conflict Graph, there has to be one malicious party which never aborts. With at least one additional party being honest, we get the bound of $(n-2)$.

Lemmas 5 to 7 play an essential role in providing a protocol for our main goal, namely SFE-expansion from $(n-1)$ parties to $n$ parties. But first, we have to show how to expand COM from $(n-1)$ parties to $n$ parties:

**Lemma 8 (Informal version of Lemma 18).** *For sufficiently small $n$ or large $t \leq (n-2)$, there exists a protocol $\pi^n_{\mathsf{COM}}$ in the $\{\mathcal{I}^{n-1}_{\mathsf{COM}}, \mathcal{I}^n_{\mathsf{BC}}\}$-hybrid model that securely UC-realizes $\mathcal{I}^n_{\mathsf{COM}}$.*

We assume that there are $(n-1)$ subfunctionalities of $\mathcal{I}^{n-1}_{\mathsf{COM}}$, one for each omitted receiver $\mathsf{R}_i$. The protocol $\pi^n_{\mathsf{COM}}$ lets the committer $\mathsf{C}$ input the same bit $b$ into all subfunctionality instances $\mathcal{I}^{n-1}_{\mathsf{COM}}$. A receiver $\mathsf{R}_i$ only accepts the global unveil of a bit $b$, if the majority of commitments that $\mathsf{R}_i$ witnesses open to $b$, if at least three instances of $\mathcal{I}^{n-1}_{\mathsf{COM}}$ were not aborted, and if at most one of the unveils opens to to a different bit. Hence, a malicious committer $\mathsf{C}$ is forced to either input the same bit into all $\mathcal{I}^{n-1}_{\mathsf{COM}}$, or to abort those where the original input differed from the to-be-unveiled bit. However, with the limit on the number of aborts before cheater identification is possible, we show in the formal proof of Lemma 18 that a corrupted $\mathsf{C}$ cannot successfully break the binding property.

With those tools, we are finally ready to investigate the expansion of FCOT. We denote by $\mathsf{S}$ the sender, by $\mathsf{R}$ the receiver and by $(\mathsf{W}_1, \ldots, \mathsf{W}_{n-2})$ the witnesses. We categorize the used subfunctionalities in two categories: *Type-1* subfunctionalities are functionalities where both $\mathsf{S}$ and $\mathsf{R}$ participate, that is, all $\mathcal{I}^{n-1}_{\mathsf{W}_i}$ for any $i \in \{1, \ldots, n-2\}$. *Type-2* subfunctionalities are $\mathcal{I}^{n-1}_{\mathsf{S}}$ and $\mathcal{I}^{n-1}_{\mathsf{R}}$.
We use Lemma 6 to limit the amount of possible aborts.

**Corollary 1.** *Let $\pi^n_{\mathsf{FCOT}}$ be a protocol that securely UC-realizes $\mathcal{I}^n_{\mathsf{FCOT}}$ in the $\{\mathcal{I}^{n-1}_{\mathsf{FCOT}}, \mathcal{I}^n_{\mathsf{CG}}\}$-hybrid model. Either at least two type-1 functionalities successfully terminate, or the deduced Conflict Graph $G^*$ is biseparated.*

This follows due to the fact that at least four subfunctionalities have to succeed. There are only two functionalities of type-2; even if those two succeed, two more have to successfully terminate, which then have to be of type-1.

We use this property to provide a protocol that proves the following lemma:

**Lemma 9 (Informal version of Lemma 19).** *For sufficiently small $n$ or large $t \leq (n-3)$, there is a protocol $\pi^n_{\mathsf{FCOT}}$ that UC-realizes $\mathcal{I}^n_{\mathsf{FCOT}}$ in the $\{\mathcal{I}^{n-1}_{\mathsf{FCOT}}, \mathcal{I}^n_{\mathsf{BC}}\}$-hybrid model against any adversary that statically corrupts at most $t \leq (n-3)$ parties.*

The actual protocol works in several iterations. Each iteration employs a cut-and-choose method, which uses one *additive* secret sharing scheme and one *threshold* secret sharing scheme: initially, $(T, r) \in \text{poly}(\lambda)$ are two natural numbers defining the total amount of shares. In each iteration, $T$ is updated as

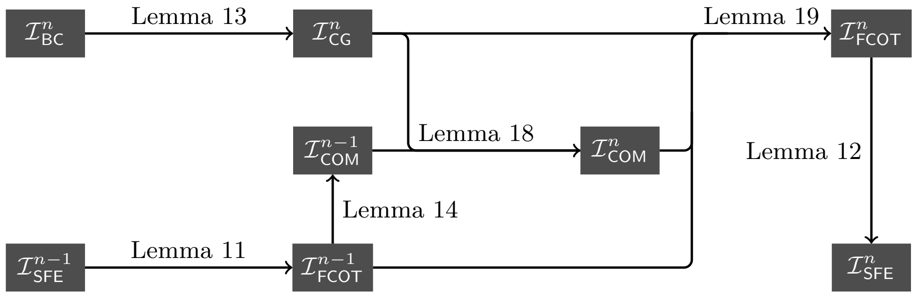

$$\begin{tikzcd}
\mathcal{I}^n_{\mathsf{BC}} \arrow[r, "\text{Lemma 13}"] & \mathcal{I}^n_{\mathsf{CG}} \arrow[rr, "\text{Lemma 19}"] \arrow[dr, "\text{Lemma 18}"] & & \mathcal{I}^n_{\mathsf{FCOT}} \arrow[d, "\text{Lemma 12}"] \\
& \mathcal{I}^{n-1}_{\mathsf{COM}} \arrow[r] & \mathcal{I}^n_{\mathsf{COM}} \arrow[ur] & \mathcal{I}^n_{\mathsf{SFE}} \\
\mathcal{I}^{n-1}_{\mathsf{SFE}} \arrow[r, "\text{Lemma 11}"] & \mathcal{I}^{n-1}_{\mathsf{FCOT}} \arrow[u, "\text{Lemma 14}"] \arrow[uurr] & &
\end{tikzcd}$$

**Fig. 3:** An overview of the steps we use for proving SFE expansion with Identifiable Abort in Theorem 8.

number of remaining subfunctionalities, which were not yet aborted. In each round, the sender uses the additive secret sharing scheme to create $T$ shares $(\mu_j^0)_{j=1..T}$ and $(\mu_j^1)_{j=1..T}$ of the messages $m_0$ and $m_1$, respectively. Let $b \in \{0,1\}$. The sender uses the $(2r/3)$-threshold secret sharing scheme to create $r$ shares $(\alpha_{j,i}^b)_{i=1..r}$ of each additive share $\mu_j^b$. Each of the obtained shares $\alpha_{j,i}^b$ is sent to $\mathcal{I}^n_{\mathsf{COM}}$; additionally, $\mathsf{S}$ distributes all the shares $\alpha_{j,i}^b$ using $(T \cdot r)$ $\mathcal{I}^{n-1}_{\mathsf{FCOT}}$ instances, with input $\big(\texttt{messages}, \alpha_{j,i}^0, \alpha_{j,i}^1\big)$. The receiver uses $\mathcal{I}^n_{\mathsf{COM}}$ to commit to the choice bit and sends the same choice bit $c$ to all instances of $\mathcal{I}^{n-1}_{\mathsf{FCOT}}$. Witnesses only output their receipt, if all type-1 functionalities $\mathcal{I}^{n-1}_{\mathsf{FCOT}}$ which have never been aborted before output a receipt.

A cut-and-choose method is used to ensure correct sharing of $\mathsf{S}$; each party broadcasts a set of $r/10n$ indices for each subfunctionality. The sender then unveils the global commitment at those spots.

This protocol allows $\mathsf{R}$ to learn $m_c$, since it can obtain sufficiently many shares to reconstruct all $T$ additive shares $\mu_j^c$. Yet it does not allow $\mathsf{R}$ to learn $m_{1-c}$; the additive secret sharing scheme used for the creation of $\mu_j^b$ ensures that the information is information-theoretically hidden if only one share is missing, and the threshold of the threshold secret sharing scheme makes it impossible for an honest receiver to obtain sufficient information to reconstruct both shares. The receiver learns its own output of $\mathcal{I}^n_{\mathsf{FCOT}}$, and $r/10$ additional shares, leaving him with $11r/10$ shares for each tuple $(\mu_j^0, \mu_j^1)$. $2r/3$ shares are required for the construction of $\mu_j^c$, yet reconstructing the whole tuple requires $4r/3$ shares.

At the same time, $\mathcal{I}^n_{\mathsf{COM}}$ ensures that both the inputs of $\mathsf{S}$ and $\mathsf{R}$ can be unveiled at a later point: A later opening of the senders inputs is trivially possible by opening all commitments on $\mu_{j,i}^b$. The other parties can check, if the unveiled values are consistent with the shares they have seen during the sending-phase. The unveil of the receivers input works directly by unveiling the choice bit from the $\mathcal{I}^{n-1}_{\mathsf{FCOT}}$ subfunctionalities.

By combining all lemmata, it follows that SFE-expansion is indeed possible in the setting of IA. A short summary of all the steps involved in the expansion alongside references to the respective proofs is given in Fig. 3.

## 2 Definitions and Notation

We provide a comprehensive glossary after our summary. Furthermore, we use the following conventions; also see the list of abbreviation and symbols.

**Notation 3** *For any natural number $n \in \mathbb{N}$, we denote by $[n]$ all natural numbers between 1 and $n$, that is,*

$$[n] \coloneqq \{1, 2, \ldots, (n-1), n\}$$

**Definition 4 (Negligible functions).** *A function $f \colon \mathbb{N} \to \mathbb{R}$ is called* negligible*, iff $f \in \mathrm{o}(x^c)$ for all $c \in \mathbb{R}$. We later use the equivalent condition $\lim\limits_{x\to\infty} \frac{\ln|f(x)|}{\ln x} = -\infty$. Denote the set of negligible functions with respect to $x$ by* $\mathrm{negl}(x)$*.*

**Definition 5 (Overwhelming functions).** *A function $f \colon \mathbb{N} \to \mathbb{R}$ is called* overwhelming*, iff $1 - f$ is negligible. Denote the set of overwhelming functions with respect to $x$ by* $\mathrm{owhl}(x)$*.*

**Notation 4 (Construction)** *We write $\mathcal{I}_A^n, \mathcal{I}_B^n \rightsquigarrow \mathcal{I}_C^n$, iff there is a protocol $\pi_{\mathsf{C}}^n$ that realizes $\mathcal{I}_C^n$ in the $\{\mathcal{I}_A^n, \mathcal{I}_B^n\}$-hybrid-model. More formally:*

$$\mathcal{I}_A^n, \mathcal{I}_B^n \rightsquigarrow \mathcal{I}_C^n \iff \exists \pi_{\mathsf{C}}^{\mathcal{I}_A^n, \mathcal{I}_B^n} : \pi_{\mathsf{C}}^{\mathcal{I}_A^n, \mathcal{I}_B^n} \geq \mathcal{I}_C^n$$

Usually the index is the security parameter $\lambda$, which we omit if it is clear from the context.

**Definition 6 (Minimal complete cardinality).** *Let $n \in \mathbb{N}$ be the number of parties and let $t \leq n$ be the maximal number of corrupted parties. We denote the cardinality of a minimal* and *complete functionality by $k^*(n,t)$. In other words, a minimal complete functionality of cardinality $m'$ is the smallest functionality from which $n$-party Secure Function Evaluation (SFE) can be constructed.*

If they are clear from the context, we omit the parameters $n$ and $t$.

We make use of secret sharing in our constructions. Thus, we include a brief description for consistency.

**Notation 5 (Secret sharing (informal))** *Let $p$ be a prime integer and let $k, m \in \mathbb{N}$ be integers such that $k \leq m \leq p$. For a secret $s \in \mathbb{Z}_p$ a secret sharing $(\sigma_i)_{i=1..m}$ is a $(k,m)$-threshold scheme, iff $k$ or more shares uniquely reconstruct the secret $s$ but $(k-1)$ or less shares hide the secret $s$ information-theoretically.*

A prominent example is Shamir's secret sharing [Sha79]. Also particularly efficient are *additive* schemes which are always $(m,m)$-threshold schemes. Here, shares are simply uniformly distributed numbers from $\mathbb{Z}_p$ such that $s = \bigoplus_{i=1}^m \sigma_i$.

**Notation 6 (Boolean random variable)** *Let $B$ be a boolean random variable. We denote the probabilities as $\Pr[B] \coloneqq \Pr[B=1]$ and $\Pr[\neg B] \coloneqq \Pr[B=0]$.*

**Definition 7 ((SFE-)Complete functionalities).** *For $n, m \in \mathbb{N}_+$, we call a functionality $\mathcal{F}^m$ of cardinality $m$* (SFE-)complete*, iff there exists a $\{\mathcal{F}^m\}$-hybrid protocol that securely realizes $\mathcal{F}^n_{\mathsf{SFE}}$ with the same abort property.*

**Definition 8 (Minimal functionality).** *We call a functionality $\mathcal{F}^m$ with cardinality $m$* minimal*, iff no functionality of lesser cardinality is complete.*

## 2.1 Functionalities

In this section, we introduce the ideal functionalities we use. We note that all following functionalities have the output property of 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. The adversary can then decide if honest parties should receive the output or not. However, in our setting of Identifiable Abort, withholding an output is only possible via *abort*, which in term identifies at least one corrupted party. The only exception we make is with respect to Broadcast, where our definition of the Conflict Graph requires some form of fairness.

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

Functionality $\mathcal{I}^n_{\mathsf{SFE}}$

$\mathcal{I}^n_{\mathsf{SFE}}$ proceeds as follows, running with security parameter $\lambda$, parties $[P] = \{\mathsf{P}_1, ..., \mathsf{P}_n\}$, input set $I \subseteq \{1, ..., n\}$, malicious parties $C \subseteq [P]$, adversary $\mathcal{A}$ and function $f\colon \left((x_i)_{i\in I}\right) \mapsto \left((y_j)_{j\in\{1,...,n\}}, \Delta\right)$ with private input $x_i$ and output $y_j$ for $\mathsf{P}_j$ and common output $\Delta$. Messages not covered here are ignored.

- When receiving $(\mathtt{input}, x_i)$ from $\mathsf{P}_i$ with $x_i \in \{0,1\}^\lambda$ and $i \in I$, store $(i, x_i)$. Ignore further messages from $\mathsf{P}_i$.
- When there are $(i, x_i)$ in store for all $i \in I$, then send $(\mathtt{output}, y_j, \Delta)$ to each party $\mathsf{P}_j$ and $(\mathtt{output})$ to $\mathcal{A}$, then terminate.

When receiving $\left(\mathtt{abort}, C'\right)$ from $\mathcal{A}$ with $C' \subseteq C$, then $\mathcal{I}^n_{\mathsf{SFE}}$ outputs $\left(\mathtt{abort}, C'\right)$ to all parties, and then terminates.

We denote by $y_i$ the private output of a party $\mathsf{P}_i$. Additionally, there is a common output $\Delta$ which is obtained by all parties. We additionally use an input set $I \subseteq [P]$ of parties which have to provide input before the computation starts. This implies that not all parties have to provide input. Otherwise, certain functionalities such as broadcast cannot be realized using SFE because receiving parties, that do not provide input, may stall the protocol execution.

$\mathcal{I}^n_{\mathsf{SFE}}$ is a versatile functionality, which can be used to realize many functionalities such as *commitments*, where only the *committer* $\mathsf{C}$ has to provide input, while the receiver $\mathsf{R}$ only obtains output.

Although $\mathcal{I}^n_{\mathsf{SFE}}$ is not well-formed, it only requires knowledge about the corrupted parties *upon abort*. In particular, functionalities with IA need to know the set of malicious parties $C \subseteq [P]$, since it must check whether the set $C'$ in the abort is indeed a subset of $C$. In this sense, our functionalities are as well-formed as possible in the setting of IA.

*Global Commitment and Broadcast* Global Commitments are an extension of two-party commitments to $n$ parties, where a single committer $\mathsf{C}$ is committed towards $(n-1)$ recipients.

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

$\mathcal{I}^n_{\mathsf{COM}}$ proceeds as follows, running with parties $[P] = \{\mathsf{C}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$, malicious parties $C \subseteq [P]$ and adversary $\mathcal{A}$. Messages not covered here are ignored.

- When receiving $(\texttt{commit}, m)$ with $m \in \{0,1\}$ from party $\mathsf{C}$, store $m$ and send $(\texttt{receipt commit})$ to all parties and to $\mathcal{A}$. Ignore further messages of the type $(\texttt{commit}, \cdot)$ from $\mathsf{C}$.
- When receiving $(\texttt{unveil})$ from party $\mathsf{C}$ and if $m$ is stored, send $(\texttt{unveil}, m)$ to all parties and to $\mathcal{A}$, then terminate.

When receiving $\big(\texttt{abort}, C'\big)$ from $\mathcal{A}$ with $C' \subseteq C$, then $\mathcal{I}^n_{\mathsf{COM}}$ outputs $\big(\texttt{abort}, C'\big)$ to all parties, and then terminates.

Beyond the inherited guarantees of two-party Commitments such as *binding* and *hiding*, $\mathcal{I}^n_{\mathsf{COM}}$ ensures *consistency* in that the committer $\mathsf{C}$ is committed to *the same* bit against all receivers.

We also make use of an ideal Broadcast functionality.

Functionality $\mathcal{I}^n_{\mathsf{BC}}$

$\mathcal{I}^n_{\mathsf{BC}}$ proceeds as follows, running with parties $[P] = \{\mathsf{S}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$, malicious parties $C \subseteq [P]$ and adversary $\mathcal{A}$. Messages not covered here are ignored.

- When receiving a message $(\texttt{input}, m)$ with $m \in \{0,1\}^*$ from party $\mathsf{S}$, send $(\texttt{output}, m)$ to all parties and to $\mathcal{A}$. Then terminate.

When receiving $\big(\texttt{abort}, C'\big)$ from $\mathcal{A}$ with $C' \subseteq C$, then $\mathcal{I}^n_{\mathsf{BC}}$ outputs $\big(\texttt{abort}, C'\big)$ to all parties, and then terminates.

Although $\mathcal{I}^n_{\mathsf{BC}}$ is a relatively weak functionality on its own, we later show that it suffices as single setup to realize the Conflict Graph functionality $\mathcal{I}^n_{\mathsf{CG}}$ from Section 3. We provide an instantiation of $\mathcal{I}^n_{\mathsf{CG}}$ using $\mathcal{I}^n_{\mathsf{BC}}$ in Section 5, thus proving that $\mathcal{I}^n_{\mathsf{CG}}$ too is a relatively weak setup.

We stress that a broadcast $\mathcal{I}^n_{\mathsf{BC}}$ can also be constructed from a global commitment $\mathcal{I}^n_{\mathsf{COM}}$, by letting the sender $\mathsf{S}$ *commit* and immediately *unveil* the message.

*Fully Committed Oblivious Transfer* Next, we introduce our novel primitive called Fully Committed Oblivious Transfer (FCOT), which we use for our construction in Section 7:

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

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

- When receiving $(\texttt{messages}, m_0, m_1)$ from $\mathsf{S}$ with $m_0, m_1 \in \{0,1\}$, store $m_0, m_1$. Ignore further messages of the type $(\texttt{messages}, \cdot, \cdot)$ from $\mathsf{S}$.
- When receiving $(\texttt{choice}, c)$ from $\mathsf{R}$ with $c \in \{0,1\}$, store $c$. Ignore further messages of the type $(\texttt{choice}, \cdot)$ from $\mathsf{R}$.
- When both messages from $\mathsf{S}$ and $\mathsf{R}$ have been received, send $(\texttt{receipt transfer}, \perp)$ to $\mathcal{A}$ and to all parties except $\mathsf{R}$, and send $(\texttt{receipt transfer}, m_c)$ to $\mathsf{R}$.
- When receiving $(\texttt{unveil message}, b)$ from $\mathsf{S}$ with $b \in \{0,1\}$ and $m_0, m_1$ are stored, send $(\texttt{unveil message}, b, m_b)$ to $\mathcal{A}$ and to all parties. Ignore further messages $(\texttt{unveil message}, b)$ from $\mathsf{S}$.
- When receiving $(\texttt{unveil choice})$ from $\mathsf{R}$ and $c$ is stored, send $(\texttt{unveil choice}, c)$ to $\mathcal{A}$ and to all parties. Ignore further messages from $\mathsf{R}$.

When receiving $\left(\texttt{abort}, C'\right)$ from $\mathcal{A}$ with $C' \subseteq C$, then $\mathcal{I}^n_{\mathsf{FCOT}}$ outputs $\left(\texttt{abort}, C'\right)$ to all parties, and then terminates.

The idea is not completely new; Crépeau [Cré90] introduced a committed variant of OT under the name of *Verifiable OT*, which was later renamed to Committed Oblivious Transfer (COT) [CvT95]. There, the sender inputs two *committed* messages $(m_0, m_1)$, and the receiver inputs the choice bit $c$. The receiver obtains the committed $m_c$ and can use this for zero-knowledge proofs, without knowing $m_{1-c}$ and without revealing $c$ in the process.

In Fully Committed Oblivious Transfer (FCOT), the sender $\mathsf{S}$ is committed to both messages $m_0$ and $m_1$, and the receiver $\mathsf{R}$ is committed to $c$. Our ideal functionality $\mathcal{I}^n_{\mathsf{FCOT}}$ is a novel extension of conventional OT, which has proven to be very useful in the setting of Identifiable Abort.

This $n$-party extension of OT is motivated by the insight that the results of [Kil88; IPS08] do not work with IA. In Section 4, we show that FCOT is SFE-complete in the setting of IA. The proof is based on the construction from [IPS08], but replaces (2-party) OT-calls with ($n$-party) FCOT-calls. When a party notices any misbehavior, it can demand other parties to open their inputs, thus enabling all parties to retrace the disruptor's misbehavior without leaking any information on the parties inputs due to their secret sharing.

## 3 Conflict Graph

We elaborate on our CG technique as sketched in Section 1.3. The main idea of the CG is that parties can publicly declare conflict with any other party if misbehavior is detected. Once sufficiently many conflicts have been declared, $n$-party Identifiable Abort, as introduced in [IOZ14], becomes possible.

To this end, we introduce an ideal functionality $\mathcal{I}^n_{\mathsf{CG}}$ which administrates observed misbehavior during the execution. In Section 5, we provide a detailed construction of $\mathcal{I}^n_{\mathsf{CG}}$ from a broadcast functionality $\mathcal{I}^n_{\mathsf{BC}}$. The functionality $\mathcal{I}^n_{\mathsf{CG}}$ abstracts away the complexity of deducing all proper conflicts from the public conflict announcements. To distinguish the Conflict Graph obtained only from *announced* conflicts and the *proper* Conflict Graph, we denote the graph produced solely by the conflict announcement as *induced* Conflict Graph by $G$, and the graph provided by $\mathcal{I}^n_{\mathsf{CG}}$ as the (deduced) Conflict Graph by $G^*$. Specifically, $\mathcal{I}^n_{\mathsf{CG}}$ applies the DeduceCG algorithm (Algorithm 1) to the induced graph. Intuitively, the DeduceCG algorithm takes the role of each party and applies a special inference rule which is best explained with an example. Consider Fig. 2c with $n = 6$ and an upper bound of malicious parties $t = 3$. This graph is neither $t$-settled nor biseparated. However, as party $\mathsf{P}_1$ has already identified its neighbors $\mathrm{N}(\mathsf{P}_1) = \{\mathsf{P}_5, \mathsf{P}_6\}$ as malicious, the remaining graph (without $\{\mathsf{P}_1, \mathsf{P}_5, \mathsf{P}_6\}$) must be explicable with $(t - |\mathrm{N}(\mathsf{P})|) = 1$ corrupted party. In this case, the remaining graph is actually 1-settled. Thus, an honest $\mathsf{P}_1$ would declare conflict with $\mathsf{P}_4$. This is exactly the conflict DeduceCG deduces, resulting in the *deduced* Conflict Graph with an extra edge $e = \{\mathsf{P}_1, \mathsf{P}_4\}$ in Fig. 2f. Note that we require a *correct* usage of $\mathcal{I}^n_{\mathsf{CG}}$, where honest parties only declare a conflict after noticing misbehavior from another party and where every noticed misbehavior is reported. This implies that each conflict must contain at least one malicious party. While honest parties only declare conflicts with *disruptors*, that is, parties that actively deviate from the protocol, malicious parties can declare arbitrary conflicts.

Let's consider another example. If a $n$-party functionality $\mathcal{I}^n$ is aborted with output $(\mathtt{abort}, \mathsf{P}_i)$, all other $(n-1)$ parties know that the corrupted party $\mathsf{P}_i$ is a disruptor. Disruptors may also notably deviate from the protocol in any protocol-dependent manner. Consider, for example, a protocol that specifies that each party initially sends the message $\mathtt{OK}$ via $\mathcal{F}^2_{\mathsf{AUTH}}$ to all other parties. If party $\mathsf{P}_j$ receives $\perp$ from $\mathsf{P}_i$ via $\mathcal{F}^2_{\mathsf{AUTH}}$, it is clear to $\mathsf{P}_j$ that $\mathsf{P}_i$ must be malicious. It is vital to ensure during protocol design, that no conflicts between two honest parties can occur due to protocol deviations. Intuitively, it must be impossible for an adversary to frame an honest party such that it is in conflict with another honest party.

The novel utility of the CG lies in the connection of (unanimous) identifiability to verifiable properties of the CG $G^*$. While formal definitions can be found in Definitions 2 and 3, here we want to go into more detail about the relation between the those graph properties. First let's briefly recall the two properties:

- $t$-settledness: Intuitively all possible explanations (MVCs) contain at least one common party. Given a $t$-settled Conflict Graph it is possible to identify at least one malicious party. In other words, participants as well as outsiders are able to identify disruptors.

- Biseparation: The graph contains two partitions that are completely in conflict with each other. Given a biseparated graph, participants can identify disruptors (but not necessarily outsiders).

*Main result for the Conflict Graph.* Next, we present the main result regarding our concept of the Conflict Graph, stating that for any protocol $\pi$ that implements a functionality with IA, a deduced biseparated Conflict Graph $G^*$ is necessary and sufficient upon abort. For the sake of brevity, when referring to the deduced Conflict Graph of a $\mathcal{I}^n_{\mathsf{CG}}$-hybrid protocol we'll simply call it the Conflict Graph.

As mentioned before, on a technical level, we only quantify over protocols that use the CG correctly. Hence we state nothing about protocols that are not in the $\mathcal{I}^n_{\mathsf{CG}}$-hybrid model. However, we show that this is *not* a restriction since each protocol can be augmented with $\mathcal{I}^n_{\mathsf{CG}}$ in a trivial manner where our result applies.

**Lemma 10.** *Let $n$ be the number of parties in $[P]$ of which at most $0 \leq t < n$ are malicious. Let $\pi$ be a protocol that securely UC-realizes a functionality $\mathcal{I}^n$ in some model $M$ excluding $\mathcal{I}^n_{\mathsf{CG}}$. Denote by $\pi'$ the $M \cup \{\mathcal{I}^n_{\mathsf{CG}}\}$-hybrid protocol that is identical to $\pi$ with the addition that it uses $\mathcal{I}^n_{\mathsf{CG}}$* correctly*. That is, whenever a subfunctionality of $M$ with parties $S \subseteq [P]$ is identifiably aborted with disruptors $D \subset S$, the honest parties in $S$ declare conflicts with $D$. Hence the subgraph on $S$ becomes biseparated.*

*Then $\pi'$ also securely UC-realizes $\mathcal{I}^n$, i.e. $\pi' \geq \mathcal{I}$.*

*Proof.* The intuition of this proof is that the information provided by $\mathcal{I}^n_{\mathsf{CG}}$ is only useful for honest parties. In other words, the environment can infer the conflict graph from its own behavior, thus $\mathcal{I}^n_{\mathsf{CG}}$ provides no advantage for the environment in distinguishing a real execution and a simulation.

Suppose for the sake of contradiction that there exists an environment $\mathcal{Z}'$ which distinguishes protocol executions of $\pi'$ from simulated ones. This environment, however, may obtain information from $\mathcal{I}^n_{\mathsf{CG}}$ by letting the dummy adversary issue a query to $\mathcal{I}^n_{\mathsf{CG}}$ in the name of any corrupted party. Hence $\mathcal{Z}'$ cannot directly be used to distinguish $\pi$ from its simulations.

Though, we can define a new environment $\mathcal{Z}$ which internally runs $\mathcal{Z}'$ and supplies it with the necessary information from a simulated $\mathcal{I}^n_{\mathsf{CG}}$. To this end $\mathcal{Z}$ start with an empty (induced) CG $G \coloneqq ([P], E)$ with $E \coloneqq \emptyset$. Whenever $\mathcal{Z}$ obtains $(\mathtt{abort}, D)$ from any corrupted party for any subfunctionality on $S \subseteq \mathsf{P}$ the environment $\mathcal{Z}$ biseparates its simulated CG $G$ on $S$ between $D$ and $S \setminus D$. When $\mathcal{Z}'$ queries $\mathcal{I}^n_{\mathsf{CG}}$ the outer environment $\mathcal{Z}$ supplies $\mathcal{Z}'$ with $G^* \coloneqq \mathrm{DeduceCG}(G, t)$. The simulated CG is identical to the one deduced by $\mathcal{I}^n_{\mathsf{CG}}$ in $\pi'$ by requirement on $\pi'$. If, for some reason, a corrupted parties declares a conflict then $\mathcal{Z}$ also incorporates this conflict into its simulated CG.

The outer environment hence distinguishes $\pi$ from its simulations in contradiction to $\pi \geq \mathcal{I}^n$. □

This preliminary result already provides a quite interesting high-level interpretation: for honest parties it is always the best strategy to immediately publicize their conflicts. In particular there is no use in deferring conflicts in an attempt to bring the adversary in more conflicts with other honest parties later on. In other words, the CG only provides a useful service to honest parties. Now that we have seen that *all* protocols can easily be augmented with $\mathcal{I}^n_{\mathsf{CG}}$, we shall proceed to show that Identifiable Abort is equivalent to biseparation of the corresponding deduced Conflict Graph.

**Theorem 7 (Biseparated Identifiable Abort).** *Let $n$ be the number of parties of which at most $0 \leq t < n$ are malicious. Let $\pi$ be a protocol that securely UC-realizes a functionality $\mathcal{I}^n$ with Identifiable Abort in an $\{\mathcal{I}^m, \mathcal{I}^n_{\mathsf{CG}}\}$-hybrid model with $m \leq n$. Furthermore let $\pi$ use $\mathcal{I}^n_{\mathsf{CG}}$ correctly, as in Lemma 10. Upon abort, the Conflict Graph $G^*$ of $\pi$ must be biseparated.*

*Proof.* The case of $n = 2$ is trivial, henceforth we assume $n \geq 3$.

We carry out our proof for Uni-Identifiable Abort, however, the same reasoning applies for Multi-Identifiable Abort. We prove that, against *some* environment $\mathcal{Z}$, a Conflict Graph that is not biseparated must lead to an abort where the identified party is honest. Together with the fact that no simulator can abort $\mathcal{I}^n$ with an honest party, this directly contradicts the presupposition that the protocol $\pi$ securely realizes $\mathcal{I}^n$.

Let $\pi$ be a protocol as described in Theorem 7 with parties $[P] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. Also, let the Conflict Graph $G^*$ of $\pi$ be **not** biseparated upon abort. For the sake of contradiction, assume that there is a *selector function* with which honest parties can still agree on a common disruptor $\mathsf{D}$. We show that this agreement cannot exist by constructing a different environment creating the same Conflict Graph transcript, but where $\mathsf{D}$ is honest.

Let $\mathrm{S}_\pi : [P] \times \tau \to 2^{[P]}$ with $(\mathsf{P}, \tau) \mapsto D$ for $D \subseteq [P]$ be the selector function that specifies the identified disruptor (set) for each (honest) party $\mathsf{P}$, given the ordered set of inputs to $\mathcal{I}^n_{\mathsf{CG}}$ as $\tau$. For the sake of simplicity, we focus here on the special case $|D| = 1$, that is, $\mathrm{S}_\pi$ outputs a single party $D = \{\mathsf{D}\}$ (Uni-Identifiable Abort). However, the proof still holds if $\mathrm{S}_\pi$ outputs $D$ with $|D| > 1$. The selector $\mathrm{S}_\pi$ must only depend on the transcript of all conflicts and the identity of the given party itself. Otherwise, the environment $\mathcal{Z}$ could induce two different abort-parties for two honest parties.

We now show that for a special environment $\mathcal{Z}^0$ there exists a different environment $\mathcal{Z}^1$ where the same transcript would lead to an accusation of an honest party. Let the first environment $\mathcal{Z}^0$ corrupt up to $t$ parties $C^0 \in M(G^*, t)$ (recall Definition 2). Denote the complement set of honest parties by $H^0 \coloneqq [P] \setminus C^0$. Environment $\mathcal{Z}^0$ produces a transcript $\tau^0$ for $\mathcal{I}^n_{\mathsf{CG}}$ until the protocol aborts with output $(\mathtt{abort}, \mathsf{D})$, where $\mathrm{S}_\pi\big(\mathsf{P}, \tau^0\big) = \{\mathsf{D}\}$ for each $\mathsf{P} \in H^0$. The environment $\mathcal{Z}^0$ only lets corrupted parties $\mathsf{D} \in C^0$ broadcast conflicts as a retaliation, that is, it broadcasts conflicts $(\mathtt{conflict}, \mathsf{P})$ against an honest party $\mathsf{P}$ in the name of $\mathsf{D}$ only after $\mathsf{P}$ has publicly declared a conflict $(\mathtt{conflict}, \mathsf{D})$. Note that this is not a restriction since we argue that there *exists* this environment $\mathcal{Z}^0$.

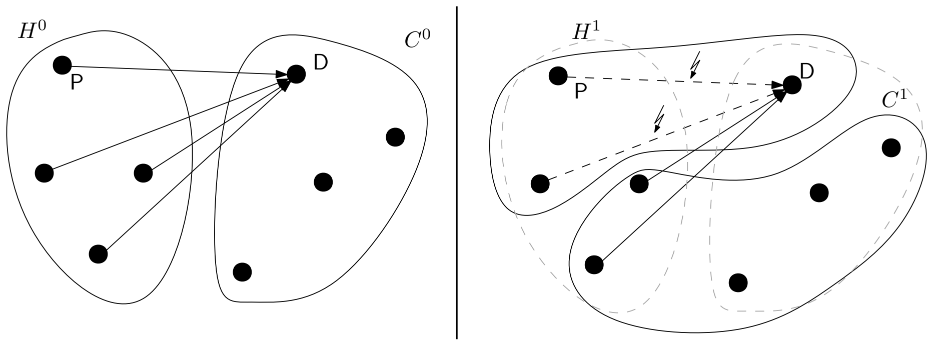


**Fig. 4:** Visualization of an exemplary set of corrupted parties and an assumed honest parties agreement on $\mathsf{D}$ for an environment $\mathcal{Z}^0$ on the left. The right side shows another, specifically constructed set of corrupted parties by an environment $\mathcal{Z}^1$ which leads to honest-honest-accusations despite having the same Conflict Graph transcript.

We now show that if there exists another corrupted set of parties (MVC) $C^1 = [P] \setminus H^1 \in M(G^*, t)$ with $C^0 \cup C^1 \neq [P]$, or equivalently $H^0 \cap H^1 \neq \emptyset$, and $\mathsf{D} \in \mathrm{S}_\pi(\mathsf{P}, \tau^0) \subseteq C^0 \cap H^1$ for all $\mathsf{P} \in H^0$, then $\pi$ cannot securely realize $\mathcal{I}^n$. Here the corrupted MVC $C^1$ is chosen such that the identified party against $\mathcal{Z}^0$ is explicitly excluded. Because there exists a *target* party $\mathsf{P}_\mathrm{t}$ in $H^0 \cap H^1$, it will select $\mathsf{P}^* \in \mathrm{S}_\pi(\mathsf{P}_\mathrm{t}, \tau^0) = \mathrm{S}_\pi(\mathsf{P}_\mathrm{t}, \tau^1)$ against both environments, though $\mathsf{D}$ is honest when playing with $\mathcal{Z}^1$. Specifically, there exists an honest party $\mathsf{P}_\mathrm{t} \in H^0 \cap H^1$ which selects $\mathrm{S}_\pi(\mathsf{P}_\mathrm{t}, \tau^0)$ against $\mathcal{Z}^0$ but selects $\mathrm{S}_\pi(\mathsf{P}_\mathrm{t}, \tau^1) \subset H^1$ against $\mathcal{Z}^1$. Since both transcripts $\tau^0$ and $\tau^1$ are equal, the two selected parties must also be equal. Fig. 4 depicts an exemplary visualization of the corruption sets.

This leads to the contradiction that against the environment $\mathcal{Z}^1$ an honest party is identified by at least one honest party. Thus, we know that if the above conditions hold, that is, there exists a MVC $C^1$ with $H^0 \cap H^1 \neq \emptyset$ and $\mathsf{D} \in C^0 \cap H^1$, then $\pi$ cannot securely realize $\mathcal{I}^n$.

Next, we show that such an MVC actually exists. It only does not exist if for all feasible explanations $C^1 \in M(G^*, t)$, it holds that $C^0 \cup C^1 = [P]$, or if for all $\mathsf{P}_\mathrm{t} \in H^0$, it holds that $\mathrm{S}_\pi(\mathsf{P}_\mathrm{t}, \tau^0) \subseteq H^0 \cup C^1$. The former is equivalent to $C^1 \supseteq H^0$. This, however, contradicts the initial assumption that $G^*$ is not biseparated. If $C^1 = H^0 = [P] \setminus C^0$ are both MVCs, this would imply a biseparated Conflict Graph with partitions $C^0$ and $C^1$. If $C^1 \supsetneq H^0$, it would follow that $H^0$ is a valid explanation, that is, $H^0 \in M(G^*, t)$, but of smaller size. Hence, we get that $C^1$ is not minimal and thus is not contained in $M(G^*, t)$. Both cases lead to a contradiction with the initial assumption.

Now that we have shown that there exists $C^1 \in M(G^*, t)$ that fulfills $C^0 \cup C^1 \neq [P]$ and $\mathrm{S}_\pi(\mathsf{P}, \tau^1) \subseteq C^0 \cap H^1$ for all $\mathsf{P} \in H^0$, we can define an environment $\mathcal{Z}^1$ that corrupts $C^1$ and acts such that it produces $\tau^1$ before the abort.

We conclude our proof by showing that $\mathcal{Z}^1$ really can create an equivalent transcript to $\tau^0$. The transcript $\tau^0$ of the Conflict Graph $G^* = ([P], E)$ is essentially an ordered list of (directed) edges $(e_j)_{j=1\dots m}$. To keep the information which party broadcasts the conflict, the edges in the transcript are directed, while the edges in the Conflict Graph are undirected. This mirrors the fact that the process of the Conflict Graph creation yields more information than the final Conflict Graph. Regardless of the environment, each edge of the transcript $e_j = (u_j, v_j)$ contains at least one corrupted party. When a conflict $e_j$ arises in the protocol execution with $\mathcal{Z}^0$, the environment $\mathcal{Z}^1$ behaves as follows:

**If $u_j \in C^0$ and $u_j \in C^1$,** $\mathcal{Z}^0$ and $\mathcal{Z}^1$ behave identically.

**If $u_j \in C^0$ and $u_j \in H^1$,** then by assumption, $\mathcal{Z}^0$ will only declare conflict $e_j$ as a retaliation, that is, after $(v_j, u_j)$ has been issued, to match the behavior of an honest party. Therefore $\mathcal{Z}^1$ lets $v_j$ broadcast $(\texttt{conflict}, u_j)$ such that the retaliation $e_j$ follows subsequently from the honest party $u_j$.

**If $u_j \in H^0$ and $u_j \in C^1$,** $\mathcal{Z}^1$ lets $u_j$ broadcast $(\texttt{conflict}, v_j)$. If $v_j$ is honest, it will retaliate; if it is not, $\mathcal{Z}^1$ lets $v_j$ broadcast the retaliation $(\texttt{conflict}, u_j)$.

**If $u_j \in H^0$ and $u_j \in H^1$,** $\mathcal{Z}^0$ and $\mathcal{Z}^1$ behave identically.

The transcript produced by $\mathcal{Z}^1$ is therefore identical to $\tau^0$, which concludes our contradiction. □

## 4 SFE-Completeness of FCOT

In this section, we prove that Fully Committed Oblivious Transfer (FCOT) and Secure Function Evaluation (SFE) are equally powerful in the setting of IA, meaning that they can be constructed from each other. This implies that constructing $n$-party SFE from $(n-1)$-party SFE and an $n$-party broadcast comes down to the more intuitive construction of $n$-party FCOT from $(n-1)$-party FCOT and a broadcast.

**Lemma 11 (SFE $\rightsquigarrow$ FCOT).** *Let $n$ be any number of parties. There is a protocol $\pi^n_{\mathsf{FCOT}}$ in the $\{\mathcal{I}^n_{\mathsf{SFE}}\}$-hybrid model that securely UC-realizes $\mathcal{I}^n_{\mathsf{FCOT}}$.*

*Proof.* We denote the set of all parties by $[P] = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, \dots, \mathsf{W}_{n-2}\}$. Further on, we use a seamless type conversion from algebraic numbers to bit strings when necessary, in the form of $\mathbb{Z}_{2^N} \cong \{0,1\}^N$.

We present a protocol that lets the sender create many secret sharings of its inputs. The receiver $\mathsf{R}$ obtains sufficiently many shares to reconstruct one message, but not enough to reconstruct both. The witnesses $\mathsf{W}_i$ obtain sufficiently many shares to detect a manipulation of the shares in the unveiling-phase, but not enough to learn any message just from the OT-phase.

First, we describe the protocol $\pi^n_{\mathsf{FCOT}}$ that utilizes four calls to $\mathcal{I}^n_{\mathsf{SFE}}$. The first instance, $\mathcal{I}^n_{\mathsf{SFE}}[f]$, is used for the OT. The next two instances $\mathcal{I}^n_{\mathsf{SFE}}[g^0_{\mathsf{S}}]$ and $\mathcal{I}^n_{\mathsf{SFE}}[g^1_{\mathsf{S}}]$ are used for the unveiling of the sender's message $m_0$ and $m_1$, respectively. The last instance $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{R}}]$ is used for the unveiling of the receiver's choice bit. The functions are defined as follows:

$$\begin{aligned} f\colon (x_{\mathsf{S}}, x_{\mathsf{R}}, x_0, ..., x_{n-3}) &\mapsto (y_{\mathsf{S}}, y_{\mathsf{R}}, y_0, ..., y_{n-3}, \Delta) \\ g^b_{\mathsf{S}}\colon \qquad\qquad (u^b_{\mathsf{S}}) &\mapsto (\varepsilon, \Delta' = u^b_{\mathsf{S}}) \\ g_{\mathsf{R}}\colon \qquad\qquad (v_{\mathsf{R}}) &\mapsto (\varepsilon, \Delta'' = v_{\mathsf{R}}) \end{aligned}$$

for both $b \in \{0,1\}$. The function $f$ provides private outputs for each party and public output $\Delta$. The inputs and outputs are defined as follows:

$$\begin{array}{rlrl} x_{\mathsf{S}} \coloneqq & \left(\left(\mu^0_{j,i}, \mu^1_{j,i}\right)_{j=1..n}, \nu^i_{\mathsf{S}}\right)_{i=1..r} & x_l \coloneqq & \left(\nu^i_l\right)_{i=1..r} \\ y_{\mathsf{S}} \coloneqq & \left(\gamma_{\nu^i_{\mathsf{S}}}\right)_{i=1..r} & y_l \coloneqq & \left(\mu^{0,i}_{\nu^i_l}, \mu^{1,i}_{\nu^i_l}, \gamma_{\nu^i_l}\right)_{i=1..r} \\ x_{\mathsf{R}} \coloneqq & \left(\left(\gamma_{j,i}\right)_{j=1..n}, \nu^i_{\mathsf{R}}\right)_{i=1..r} & u^b_{\mathsf{S}} \coloneqq & \left(\mu^b_{j,i}\right)_{j=1..n, i=1..r} \\ & & \Delta \coloneqq & (\Delta_{\mathsf{S}}, \Delta_{\mathsf{R}}) \\ y_{\mathsf{R}} \coloneqq & \left(m_c, \left(\mu^{0,i}_{\nu^i_{\mathsf{R}}}, \mu^{1,i}_{\nu^i_{\mathsf{R}}}\right)_{i=1..r}\right) & v_{\mathsf{R}} \coloneqq & \left(\gamma_{j,i}\right)_{j=1..n, i=1..r} \end{array}$$

Let $r \in \omega\left(n^2 + n \ln \lambda\right)$ be the number of shares in which each $\gamma_{j,i}$ is divided. For $i \in [r]$, the sharings $\bigoplus_{j=1}^n \mu^0_{j,i} \eqqcolon m_0$ and $\bigoplus_{j=1}^n \mu^1_{j,i} \eqqcolon m_1$ distribute the two messages $m_0$ and $m_1$, and the sharing $\bigoplus_{j=1}^n \gamma_{j,i} \coloneqq c$ distributes the choice bit $c$. The number of sharings $r$ is chosen such that the detection of a share alteration by all honest parties becomes overwhelming. The common output $\Delta_{\mathsf{S}}$ takes the value 1, if the encoded bits of all $r$ sharings of $m_0$ are equal ($m_0 = \bigoplus_{j=1}^n \mu^0_{j,i}$ for all $i \in [r]$) and all encoded bits of the sharings of $m_1$ are equal. Otherwise $\Delta_{\mathsf{S}}$ is 0. Analogously, $\Delta_{\mathsf{R}}$ is 1, if the sharings of $c$ are consistent and 0 otherwise. Formally, the shares are bits $\mu^0_{j,i}, \mu^1_{j,i}, \gamma_{j,i} \in \{0,1\}$ and the share choices are numbers $\nu^i \in \mathbb{Z}_n$. Now that we have the definitions of the SFE-subfunctionalities we proceed to describe the actual protocol. We assume inputs $\mathsf{S}(m_0, m_1)$, $\mathsf{R}(c)$ and $\mathsf{W}_l(\varepsilon)$ with empty string $\varepsilon$. Additionally, each party implicitly obtains a unary representation of the security parameter $1^\lambda$.

The protocol looks as follows:

**On input** (`local start`) from $\mathcal{Z}$ to $\mathsf{W}_l$, $\mathsf{W}_l$ draws $r$ numbers $\left\{\nu^i_l\right\}_{i\in[r]} \overset{\$}{\leftarrow} \mathbb{Z}^r_n$, to determine which share of the other parties' inputs will be obtained by $\mathsf{W}_l$.

**On input** ($\texttt{messages}, m_0, m_1$) from $\mathcal{Z}$ to $\mathsf{S}$, $\mathsf{S}$ produces $r$ independent, additive $n$-sharings of $m_0$ and $m_1$: $(\mu^0_{j,i}, \mu^1_{j,i})_{j\in[n], i\in[r]} \overset{\$}{\leftarrow} \mathbb{Z}^{2r\cdot n}_n$ such that for all $i \in [r]$ it holds that $\bigoplus_{j\in[n]} \mu^0_{j,i} = m_0$ and $\bigoplus_{j\in[n]} \mu^1_{j,i} = m_1$, where $j$ is the index of the share within a sharing and $i$ is the index of the sharing itself. Then, $\mathsf{S}$ draws $r$ numbers $\left\{\nu^i_{\mathsf{S}}\right\}_{i\in[r]} \overset{\$}{\leftarrow} \mathbb{Z}^r_n$, which determine the shares of the receiver's choice bit $\mathsf{S}$ obtains. $\mathsf{S}$ inputs $x_{\mathsf{S}}$ into $\mathcal{I}^n_{\mathsf{SFE}}[f]$.

**On input** ($\texttt{choice}, c$) from $\mathcal{Z}$ to $\mathsf{R}$, $\mathsf{R}$ produces $r$ independent, additive $n$-sharings of $c$: $(\gamma_{j,i})_{j=1\in[n], i\in[r]} \overset{\$}{\leftarrow} \mathbb{Z}^{r\cdot n}_n$ such that for all $i \in [r]$ it holds that $\bigoplus_{j=1\in[n]} \gamma_{j,i} = c$, where $j$ is the index of the share within a sharing and $i$ is the index of the sharing itself. Then, $\mathsf{R}$ draws $r$ numbers $\left\{\nu^i_{\mathsf{R}}\right\}_{i\in[r]} \overset{\$}{\leftarrow} \mathbb{Z}^r_n$, which determine the shares of the sender's input $\mathsf{R}$ obtains. $\mathsf{R}$ inputs $x_{\mathsf{R}}$ into $\mathcal{I}^n_{\mathsf{SFE}}[f]$.

**On output** $\mathit{\Delta}_{\mathsf{S}} = 0$ or $\mathit{\Delta}_{\mathsf{R}} = 0$ from $\mathcal{I}^n_{\mathsf{SFE}}[f]$ to $\mathsf{P}$, where $\mathsf{P} \in [P]$, the party $\mathsf{P}$ aborts with output $\big(\texttt{abort}, C'\big)$, where $\mathsf{S} \in C' \iff \mathit{\Delta}_{\mathsf{S}} = 0$ and $\mathsf{R} \in C' \iff \mathit{\Delta}_{\mathsf{R}} = 0$.

**On output** $(\texttt{output}, y_{\mathsf{R}}, \mathit{\Delta})$ from $\mathcal{I}^n_{\mathsf{SFE}}[f]$ to $\mathsf{R}$, $\mathsf{R}$ outputs $(\texttt{receipt transfer}, m_c)$.

**On output** $(\texttt{output}, y_{\mathsf{P}}, \mathit{\Delta})$ from $\mathcal{I}^n_{\mathsf{SFE}}[f]$ to $\mathsf{P}$, where $\mathsf{P} \in [P] \setminus \mathsf{R}$, party $\mathsf{P}$ outputs $(\texttt{receipt transfer}, \bot)$.

**On input** $(\texttt{unveil message}, b)$ from $\mathcal{Z}$ to $\mathsf{S}$, if $(\texttt{messages}, m_0, m_1)$ has been received, $\mathsf{S}$ inputs the previously generated $u^b_{\mathsf{S}}$ into $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$.

**On output** $u^b_{\mathsf{S}}$ from $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$ to $\mathsf{P}$, where $\mathsf{P} \in [P] \setminus \mathsf{S}$, $\mathsf{P}$ checks the consistency with all previously obtained shares. If all shares match and are consistent, i.e. across all $i \in [r]$ the sharings encode the same bit, $\mathsf{P}$ outputs $(\texttt{unveil message}, b, m_b)$. Otherwise, $\mathsf{P}$ outputs $(\texttt{abort}, \mathsf{S})$.

**On input** $(\texttt{unveil choice})$ from $\mathcal{Z}$ to $\mathsf{R}$, if $(\texttt{choice}, c)$ has been received, $\mathsf{R}$ inputs $v_{\mathsf{R}}$ into $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{R}}]$.

**On output** $v_{\mathsf{R}}$ from $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{R}}]$ to $\mathsf{P}$, where $\mathsf{P} \in [P] \setminus \mathsf{R}$, $\mathsf{P}$ checks the consistency with their previously obtained shares. If all shares match and are consistent across $i \in [r]$, $\mathsf{P}$ outputs $(\texttt{unveil choice}, c)$. Otherwise, $\mathsf{P}$ outputs $(\texttt{abort}, \mathsf{R})$.

We now give a description of the simulator. At the onset of the simulation, the simulator follows the program of all uncorrupted witnesses on input $(\texttt{local start})$ and computes their input $x_l$ according to the protocol and inputs it into the simulated $\mathcal{I}^n_{\mathsf{SFE}}$ in the name of $\mathsf{W}_l$.

**On input** $(\texttt{receipt messages}, \bot)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$, the simulator gives local input to all uncorrupted simulated parties as follows:
The simulator gives the simulated $\mathsf{S}$ local input $(\texttt{messages}, 0, 0)$. Thus, $\mathsf{S}$ computes $x_{\mathsf{S}}$ according to the protocol and sends it to $\mathcal{I}^n_{\mathsf{SFE}}[f]$.
The simulator gives the simulated $\mathsf{R}$ local input $(\texttt{choice}, 0)$. Thus, $\mathsf{R}$ computes $x_{\mathsf{R}}$ according to the protocol and sends it to $\mathcal{I}^n_{\mathsf{SFE}}[f]$.

**On output** $(\texttt{receipt output})$ from $\mathcal{I}^n_{\mathsf{SFE}}[f]$, the simulator reports $(\texttt{receipt output})$ to $\mathcal{Z}$.
If the common output of $\mathcal{I}^n_{\mathsf{SFE}}[f]$ contains $\mathit{\Delta}_{\mathsf{S}} = 0$ or $\mathit{\Delta}_{\mathsf{R}} = 0$, the simulator aborts $\mathcal{I}^n_{\mathsf{FCOT}}$ with output $\big(\texttt{abort}, C'\big)$ where $\mathsf{S} \in C'$ or $\mathsf{R} \in C'$.

**On output** $(\texttt{unveil message}, b, m_b)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$, the simulator fabricates input $u^b_{\mathsf{S}}$ for the uncorrupted simulated $\mathsf{S}$ to send as input into $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$. The simulator knows the indices of all shares of $\mu^0$ and $\mu^1$ that have been chosen by all other parties in the OT-phase. Either $\mathcal{S}$ learned the choice indices from a corrupted party as local input or it generated the choice indices itself for the uncorrupted simulated parties. Because each sharing has $n$ additive shares, there is at least one index ${\nu'}^i \in \mathbb{Z}_n$ for each $i \in \{1, ..., r\}$ whose share is known by no party (recall that only $(n-1)$ shares per sharing have been distributed). Hence, the simulator flips the bits at the correct index for each $i \in [r]$ if necessary, that is, iff $m_b = 1$; recall that $\mu^0$ and $\mu^1$ encode 0. Denote the manipulated sharings by ${\mu'}^0$ and ${\mu'}^1$; they encode $m_0$ resp. $m_1$. Finally, the simulator lets $\mathsf{S}$ input ${u'}^b_{\mathsf{S}}$ (computed from ${\mu'}^b$) into $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$.

Because the manipulations of the sharings are chosen specifically such that no party yields a share that is manipulated, no party will notice the equivocation and accept the unveiling by outputting $(\texttt{unveil message}, b, m_b)$.

**On output** $(\texttt{unveil choice}, c)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$, the simulator fabricates input $u_{\mathsf{R}}$ for the uncorrupted simulated $\mathsf{R}$ to input into $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{R}}]$. Here, the simulator proceeds completely analogous to the previous case of the unveiling of the sender's messages. The same argumentation regarding the acceptance of the fabricated sharing applies.

**On input** $\big(\texttt{abort}, C'\big)$ for $\mathcal{I}^n_{\mathsf{SFE}}[\cdot]$, the simulator aborts $\mathcal{I}^n_{\mathsf{SFE}}[\cdot]$ as well as $\mathcal{I}^n_{\mathsf{FCOT}}$ with $\big(\texttt{abort}, C'\big)$.

Finally, we describe the simulator's behavior when handling messages from and to malicious parties. In general, messages between hybrid functionalities and the environment are forwarded by the simulator. For simplicity, we assume $r$ to be an *uneven* number.

**On input** $(\texttt{input}, x_{\mathsf{S}})$ from corrupted $\mathsf{S}$ to $\mathcal{I}^n_{\mathsf{SFE}}[f]$, the simulator lets $\mathsf{S}$ input $(\texttt{messages}, m_0, m_1)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$, where $m_0$ and $m_1$ are the respective majority of the $r$ encoded bits $m^{b,i} = \bigoplus_{j=1}^{n} \mu^b_{j,i}$. The input $(\texttt{input}, x_{\mathsf{S}})$ is naturally passed on to $\mathcal{I}^n_{\mathsf{SFE}}[f]$.

**On input** $(\texttt{input}, x_{\mathsf{R}})$ from corrupted $\mathsf{R}$ to $\mathcal{I}^n_{\mathsf{SFE}}[f]$, the simulator lets $\mathsf{R}$ input $(\texttt{choice}, c)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$, where $c$ is the majority of the encoded bits $c^i = \bigoplus_{j=1}^{n} \gamma_{j,i}$. The input $(\texttt{input}, x_{\mathsf{R}})$ is naturally passed on to $\mathcal{I}^n_{\mathsf{SFE}}[f]$.

**On input** $\big(\texttt{input}, u^b_{\mathsf{S}}\big)$ from corrupted $\mathsf{S}$ to $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$, the simulator checks consistency of the sharings in $u^b_{\mathsf{S}}$ with the sharings in $x_{\mathsf{S}}$. If they are consistent, then the simulator lets $\mathsf{S}$ input $(\texttt{unveil message}, b, m_b)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$, otherwise the simulator aborts $\mathcal{I}^n_{\mathsf{FCOT}}$ with output $(\texttt{abort}, \mathsf{S})$.

**On input** $(\texttt{input}, u_{\mathsf{R}})$ from corrupted $\mathsf{R}$ to $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{R}}]$, the simulator checks consistency of the sharings in $v_{\mathsf{R}}$ with the sharings in $x_{\mathsf{R}}$. If they are consistent, then the simulator lets $\mathsf{R}$ input $(\texttt{unveil choice})$ into $\mathcal{I}^n_{\mathsf{FCOT}}$, otherwise the simulator aborts $\mathcal{I}^n_{\mathsf{FCOT}}$ with output $(\texttt{abort}, \mathsf{R})$.

It remains to be shown that the simulator does provide an indistinguishable view for any input of $\mathcal{Z}$. If inconsistent sharings for $m_0$ are fed into $\mathcal{I}^n_{\mathsf{SFE}}[f]$, then, upon termination of $\mathcal{I}^n_{\mathsf{SFE}}[f]$, all honest parties certainly abort with output $(\texttt{abort}, \mathsf{S})$ due to the common output $\varDelta$. Also, if inconsistent sharings for $m_0$ are unveiled in $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{S}}]$, then all honest parties certainly abort with output $(\texttt{abort}, \mathsf{S})$, as all parties can check their consistency themselves. Consequently, a discrimination between the hybrid and ideal execution can only occur when the input sharings and the unveiled sharings are (internally) consistent but unequal. Fortunately, in this case all parties notice (with overwhelming probability) that the original input $m'_0$ and the unveiled value $m_0$ are not equal. This can be shown as follows: First note that in each of the $r$ sharings at least one share must have been altered, otherwise the sharings are inconsistent. We denote this fact as $Z \geq 1$ to indicate at least one alteration in each sharing. Now, let $H^i_{\mathsf{P}}$ be a boolean random variable and $H^i_{\mathsf{P}} = 1$ be the event that the $i$-th share index of (honest) party $\mathsf{P}$ matches the index of the maliciously altered share, else $H^i_{\mathsf{P}} = 0$. Using the our Notation 6, we get the probability $\Pr\big[H^i_{\mathsf{P}} \mid Z \geq 1\big] \geq 1/n$. Let

$N_{\mathsf{P}} \coloneqq \bigvee_{i=1}^{r} H_{\mathsf{P}}^{i} = \neg\bigwedge_{i=1}^{r} \neg H_{\mathsf{P}}^{i}$ be the boolean random variable that describes whether $\mathsf{P}$ notices an alteration in any of the $r$ sharings. We provide an upper bound for the probability $N_{\mathsf{P}} = 1$ by

$$\begin{aligned}
\Pr[N_{\mathsf{P}} \mid Z \geq 1] &= 1 - \Pr[\neg N_{\mathsf{P}} \mid Z \geq 1] \\
&= 1 - \Pr\left[\bigwedge_{i=1}^{r} \neg H_{\mathsf{P}}^{i} \,\middle|\, Z \geq 1\right] \\
&= 1 - \prod_{i=1}^{r} \Pr\left[\neg H_{\mathsf{P}}^{i} \mid Z \geq 1\right] \\
&\overset{(1.1)}{\geq} 1 - \prod_{i=1}^{r} (1 - 1/n) \\
&= 1 - (1 - 1/n)^{r}
\end{aligned} \tag{1}$$

where (1.1) uses the fact that the different $H_{\mathsf{P}}^{i}$ are independent of each other. Let the number of honest parties be $h$. The final probability $p$ that all honest parties notice any fault is then greater than $(1 - (1 - 1/n)^{r})^{h}$ which can be bounded by $p > (1 - (1 - 1/n)^{r})^{n}$. We apply some transformations to the desired condition $p \in \mathrm{owhl}(\lambda)$ and get the sufficient condition

$$\begin{aligned}
1 - p &\leq 1 - (1 - (1 - 1/n)^{r})^{n} \\
&= 1 - \sum_{i=0}^{n} \binom{n}{i} (-1)^{i} (1 - 1/n)^{ri} \\
&= \sum_{i=1}^{n} \binom{n}{i} (-1)^{i+1} (1 - 1/n)^{ri} \\
&\leq \left| \sum_{i=1}^{n} \binom{n}{i} (-1)^{i+1} (1 - 1/n)^{ri} \right| \\
&\leq \sum_{i=1}^{n} \binom{n}{i} \left| (-1)^{i+1} (1 - 1/n)^{ri} \right| \\
&= \sum_{i=1}^{n} \binom{n}{i} (1 - 1/n)^{ri} \\
&\leq \binom{n}{n/2} \sum_{i=1}^{n} (1 - 1/n)^{ri} \\
&\overset{(2.1)}{=} \binom{n}{n/2} (1 - 1/n)^{r} \cdot \frac{((1 - 1/n)^{r})^{n} - 1}{(1 - 1/n)^{r} - 1} \\
&\leq \binom{n}{n/2} (1 - 1/n)^{r} \\
&\overset{!}{\in} \mathrm{negl}(\lambda)
\end{aligned} \tag{2}$$

where (2.1) follows from the partial geometric sum with $q = (1 - 1/n)^r$. Since $\binom{n}{n/2} \in \Theta(2^n/\sqrt{n})$ the condition $\binom{n}{n/2}(1 - 1/n)^r \in \operatorname{negl}(\lambda)$ is in particular fulfilled by $r \in \omega\big(n^2 + n \ln \lambda\big)$. We write $r = n(n + \ln \lambda)w$ with $w \in \omega(1)$ with respect to $\lambda \to \infty$, thus the above claim follows from

$$\begin{aligned} 2^n/\sqrt{n} \cdot (1 - 1/n)^{n(n+\ln \lambda)w} &\leq 2^n \cdot (1 - 1/n)^{n(n+\ln \lambda)w} \\ &= 2^n \cdot \big((1 - 1/n)^n\big)^{(n+\ln \lambda)w} \\ &\leq 2^n \cdot (1/e)^{(n+\ln \lambda)w} \\ &= 2^n \cdot e^{-nw} \lambda^{-w} \\ &\leq \lambda^{-w} \\ &\in \operatorname{negl}(\lambda) \ . \end{aligned} \tag{3}$$

Although our bound for $r$ is probably not tight, we already see that the protocol is at most quadratic in $n$ and logarithmic in $\lambda$.

□

**Lemma 12 (FCOT $\rightsquigarrow$ SFE).** *Let $n$ be any number of parties. There is a protocol $\pi^n_{\mathsf{SFE}}$ in the $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$-hybrid model that securely UC-realizes $\mathcal{I}^n_{\mathsf{SFE}}$.*

*Proof.* Here, we prove that there exists a $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$- hybrid protocol $\varPhi$ that securely UC-realizes $\mathcal{I}^n_{\mathsf{SFE}}$. Again, for arbitrary $n$, denote the set of parties by $[P] = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, \dots, \mathsf{W}_{n-2}\}$.

We use the IPS-compiler from Ishai, Prabhakaran, and Sahai [IPS08], which compiles two protocols $\varPi$ and $\rho$ into an $\big\{\mathcal{F}^2_{\mathsf{OT}}\big\}$-hybrid protocol $\varPhi$. There, the outer protocol $\varPi$ can be formulated in the client-server model and must be secure against a constant fraction of malicious parties, say $t \leq n/4$. The inner protocol $\rho$ needs to be secure against arbitrarily many, semi-honest (passive) corruptions; it may be in the $\big\{\mathcal{F}^2_{\mathsf{OT}}\big\}$-hybrid model. Both protocols depend on the actual function $f$ that is to be evaluated. The combined protocol $\varPhi$ then securely realizes $\mathcal{I}^n_{\mathsf{SFE}}$ with Anonymous Abort, iff the outer protocol $\varPi$ securely realizes $\mathcal{I}^n_{\mathsf{SFE}}$. This holds both in the computational and the statistical case.

Their result cannot be directly transferred into the setting of IA. However, if we replace $\mathcal{F}^2_{\mathsf{OT}}$-calls with to $\mathcal{I}^n_{\mathsf{FCOT}}$, we claim that their result still holds. When a party $\mathsf{P}$ notices misbehavior in server $i$, it can publicly demand the unveiling of all communication corresponding to server $i$. If any party refuses to unveil, it must be malicious and all honest parties can abort with said party. If all inputs into server $i$ have been unveiled, then either all parties can retrace any deviation from the correct protocol transcript, or no actual misbehavior has occurred, then the initial party $\mathsf{P}$ demanded the unveiling without justification. Either way, at least one party can be identified.

Additionally, we must ensure that unveiling all inputs of a single server does not compromise the privacy to the inputs of the outer protocol $\varPi$. In the following, we formalize this idea: Let $n$ be the number of parties (clients) of the outer protocol. In the original paper [IPS08] there are $m \in \Theta\big(n^2\lambda\big)$ 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 unveiled 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.

To make this more formal, we consider the function $f$ that presents a boolean circuit in $\mathrm{NC}^1$. Then, using the BGW-protocol [BGW88] for $\Pi$ and the GMW-protocol [GMW87] for $\rho$, we obtain a protocol $\Phi$ that securely realizes $\mathcal{F}^n_{\mathsf{SFE}}[f]$ against arbitrarily many corruptions with Anonymous Abort. Now, we replace all calls to $\mathcal{F}^2_{\mathsf{OT}}$ with calls to $\mathcal{I}^n_{\mathsf{FCOT}}$. OT-calls are made in the distribution phase of the watchlist mechanism and in the inner protocol, in particular the outer protocol stays unchanged. Call the new protocols $\Phi'$ and $\rho'$. We require that all communication in the new inner protocol is processed via FCOTs. This is not an additional restriction because a secure-channel can be trivially realized by OT and thus by FCOT. However, it enables us to unveil all communication of the inner protocol upon abort.

If not aborted, the original protocol $\Phi$ and the new protocol $\Phi'$ yield exactly the same results. Note that the original simulator $\mathcal{S}$ and the new simulator $\mathcal{S}'$ learn exactly the same information, if no abort occurs.

In the original protocol, if any party aborts, then the original simulator $\mathcal{S}$ also abort the ideal functionality $\mathcal{F}^n_{\mathsf{SFE}}[f]$. The new simulator $\mathcal{S}'$, however, must provide a set of corrupted parties to abort the ideal functionality $\mathcal{I}^n_{\mathsf{SFE}}[f]$. Hence, all simulated parties in the simulated new protocol $\Phi'$ must provide output $\big(\mathtt{abort}, C'\big)$. The new protocol ensures this in the following way. Whenever a party $\mathsf{P}$ aborts in the original protocol $\Phi$ due to a malicious message from the $i$-th server, it, instead, broadcasts $(\mathtt{challenge}, i)$. Then, all parties unveil the FCOTs used to distribute their watchlist one-time pads and all FCOTs in the $i$-th instance of the inner protocol ($i$-th server). More precisely, we actually assume a $\binom{n^2\lambda}{\lambda}$-FCOT 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 unveiled.

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. If the aborting party $\mathsf{P}$ indeed received a malicious message on the $i$-th server, then all party notice this misbehavior and identify the disruptor party $\mathsf{P}$. They then abort with $\big(\mathtt{abort}, \mathsf{P}'\big)$. If the aborting party lied about receiving a malicious message on the $i$-th server, then the other parties can retrace that all message that $\mathsf{P}$ received were indeed correct, and they will abort with $(\mathtt{abort}, \mathsf{P})$. Either way, the simulator will abort the ideal functionality $\mathcal{I}^n_{\mathsf{SFE}}$ with the corresponding abort output.

Note that because the abort happens at exactly the same time as in the original protocol, the new protocol leaks exactly as much information as the original one which is secure. Also, by unveiling the state of the corrupted server $i$, the adversary does not learn anything that it did not know beforehand. $\square$

## 5 Conflict Graph from Broadcast

In this section, we present a protocol $\pi^n_{\mathsf{CG}}$, that realizes $\mathcal{I}^n_{\mathsf{CG}}$ in the $\{\mathcal{I}^n_{\mathsf{BC}}\}$-hybrid model, which proves the following lemma:

**Lemma 13.** *Let $n$ be the number of parties of which at most $0 \leq t < n$ are malicious; subject to $n \in \mathcal{O}(\ln \lambda) \vee n - t \in \mathcal{O}(1)$. There is a protocol $\pi^n_{\mathsf{CG}}$ in the $\{\mathcal{I}^n_{\mathsf{BC}}\}$-hybrid model that securely UC-realizes $\mathcal{I}^n_{\mathsf{CG}}$:*

$$\mathcal{I}^n_{\mathsf{BC}} \rightsquigarrow \mathcal{I}^n_{\mathsf{CG}} \tag{4}$$

*Proof.* We proof our statement by providing a protocol description for $\pi^n_{\mathsf{CG}}$ and prove it secure by providing a simulator. We have $n$ parties $[P] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. The protocol is given as follows:

**Initialize.** All parties start with a graph $G = ([P], E)$ with $E = \emptyset$.

**On input** $(\texttt{conflict}, \mathsf{P}_i)$ from $\mathcal{Z}$ to $\mathsf{P}_j$, $\mathsf{P}_j$ inputs $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ to $\mathcal{I}^n_{\mathsf{BC}}$.

**On input** $(\texttt{output}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ from $\mathcal{I}^n_{\mathsf{BC}}$, all parties $\mathsf{P} \in [P]$ add $\{\mathsf{P}_j, \mathsf{P}_i\}$ to $E$.

**On input** $(\texttt{query})$ from $\mathcal{Z}$ to $\mathsf{P}_i$, $\mathsf{P}_i$ locally computes $G^* \coloneqq \mathrm{DeduceCG}(G, t)$ and outputs $G^*$.

A simulator for this protocol is straightforward:

1. If $\mathsf{P}$ is corrupted:
   On input $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ from $\mathsf{P}$ to $\mathcal{I}^n_{\mathsf{BC}}$, if $\mathsf{P}_j = \mathsf{P}$, $\mathcal{S}$ calls $\mathcal{I}^n_{\mathsf{CG}}$ with input $(\texttt{conflict}, \mathsf{P}_i)$ in the name of $\mathsf{P}$ and sends $(\texttt{output}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ to all other parties in the name of $\mathcal{I}^n_{\mathsf{BC}}$.
2. If $\mathsf{P}$ is honest:
   On input $(\texttt{conflict}, \mathsf{P}, \mathsf{P}_i)$ from $\mathcal{I}^n_{\mathsf{CG}}$ to $\mathcal{S}$, $\mathcal{S}$ calls $\mathcal{I}^n_{\mathsf{BC}}$ with input $(\texttt{output}, (\texttt{conflict}, \mathsf{P}, \mathsf{P}_i))$ into $\mathcal{I}^n_{\mathsf{BC}}$ in the name of $\mathsf{P}$.

The simulator provides an indistinguishable view for $\mathcal{Z}$:

- Inputs $(\texttt{query})$ do not have to be handled at all. For honest parties, the parties merely forward the request and obtain the correct conflict graph $G^*$. Corrupted parties neither send messages for $(\texttt{query})$, nor change the state of the functionality $\mathcal{I}^n_{\mathsf{CG}}$ in any way, since $G^*$ is computed locally in the protocol.
- For corrupted parties, $\mathcal{S}$ obtains the input via the simulated $\mathcal{I}^n_{\mathsf{BC}}$. If the broadcasted message was valid, $\mathcal{S}$ inputs this into $\mathcal{I}^n_{\mathsf{CG}}$, thus causing the same behavior as if an honest party had called $\mathcal{I}^n_{\mathsf{CG}}$.
- For honest parties, $\mathcal{S}$ only has to simulate the behavior of $\mathcal{I}^n_{\mathsf{BC}}$.

The restriction $n \in \mathcal{O}(\ln \lambda) \vee n - t \in \mathcal{O}(1)$ comes from the fact that parties must compute DeduceCG. $\square$

## 6 Global Commitment from Fully Committed Oblivious Transfer

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

**Lemma 14.** *There is a protocol $\pi^n_{\mathsf{COM}}$ that securely UC-realizes $\mathcal{I}^n_{\mathsf{COM}}$ in the $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$-hybrid model:*

$$\mathcal{I}^n_{\mathsf{FCOT}} \rightsquigarrow \mathcal{I}^n_{\mathsf{COM}}$$

*Proof.* We assume our $n$ parties for $\mathcal{I}^n_{\mathsf{COM}}$ to be $[P] \coloneqq (\mathsf{C}, \mathsf{R}_1, \dots, \mathsf{R}_{n-1})$, our $n$ parties for $\mathcal{I}^n_{\mathsf{FCOT}}$ are $[P]' \coloneqq (\mathsf{S}, \mathsf{R}, \mathsf{W}_1, \dots, \mathsf{W}_{n-2})$. We start by sketching the protocol:

**On input** $(\texttt{commit}, m)$ for $m \in \{0,1\}$ from $\mathcal{Z}$ to $\mathsf{C}$, $\mathsf{C}$ acts as $\mathsf{R}$ in $\mathcal{I}^n_{\mathsf{FCOT}}$ and inputs $(\texttt{choice}, m)$.

**On input** $(\texttt{receipt commit})$ from $\mathcal{I}^n_{\mathsf{COM}}$ to $\mathsf{R}_i$ for $i \in [n-1]$, all receiver for $i \neq 1$ ignore the message. $\mathsf{R}_1$ acts as the sender in $\mathcal{I}^n_{\mathsf{FCOT}}$: it draws one random message $m' \xleftarrow{\$} \{0,1\}$ and sends $(\texttt{messages}, m', m')$ to $\mathcal{I}^n_{\mathsf{FCOT}}$.

**On input** $(\texttt{unveil})$ from $\mathcal{Z}$ to $\mathsf{C}$ and $(\texttt{receipt transfer}, m_c)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$ to $\mathsf{C}$, $\mathsf{C}$ sends $(\texttt{unveil choice})$ to $\mathcal{I}^n_{\mathsf{FCOT}}$.

**On input** $(\texttt{unveil choice}, c)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$ to any receiver $\mathsf{R}_i$ for $i \in [n-1]$, $\mathsf{R}_i$ outputs $(\texttt{output}, c)$.

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

1. If $\mathsf{C}$ is corrupted:
   On input $(\texttt{choice}, c)$ from $\mathsf{C}$ to $\mathcal{I}^n_{\mathsf{FCOT}}$, $\mathcal{S}$ sends $(\texttt{commit}, c)$ to $\mathcal{I}^n_{\mathsf{COM}}$ in the name of $\mathsf{C}$.
2. If $\mathsf{C}$ is honest:
   On input $(\texttt{receipt commit})$ from $\mathcal{I}^n_{\mathsf{COM}}$, $\mathcal{S}$ simulates $\mathcal{I}^n_{\mathsf{FCOT}}$ according to the code of $\mathcal{S}_{\text{FCOT}}$ with arbitrary input.
3. If $\mathsf{R}_i$ for $i \in [n]$ is corrupted:
   On input $(\texttt{messages}, m_0, m_1)$, if $m_c \notin \{0,1\}$, $\mathcal{S}$ aborts with output $\mathsf{R}_1$. Else, $\mathcal{S}$ reports $(\texttt{receipt transfer})$ to $\mathcal{Z}$.
4. If $\mathsf{R}_i$ for $i \in [n]$ is honest:
   $\mathcal{S}$ acts according to the protocol of $\mathsf{R}_i$.
5. If $\mathsf{C}$ is corrupted:
   On input $(\texttt{unveil choice})$ from $\mathsf{R}$ to $\mathcal{I}^n_{\mathsf{FCOT}}$, $\mathcal{S}$ sends $c$ to all $\mathsf{R}_i$ for $i \in [n-1]$.
6. If $\mathsf{C}$ is honest:
   On input $(\texttt{unveil}, c)$ from $\mathcal{I}^n_{\mathsf{COM}}$, $\mathcal{S}$ reports message $(\texttt{unveil choice}, c)$ to all $\mathsf{R}_i$.

The simulator trivially provides an indistinguishable view:

- Simulation of $\mathcal{I}^n_{\mathsf{FCOT}}$ follows from simulation-based security.
- The only secret is the to-be-committed bit $m$, as the receivers $\mathsf{R}_i$ obtain no secret input, meaning that $\mathcal{S}$ can execute their protocol.
- Against an honest committer, $\mathcal{S}$ just has to send messages from $\mathcal{I}^n_{\mathsf{FCOT}}$ accordingly and pretend that $\mathsf{C}$ used the correct choice bit – which does not

have to be known in advance, as $(\texttt{unveil choice}, c)$ is only required *after* $\mathcal{I}^n_{\mathsf{COM}}$ unveiled $c$.

- Against a corrupted committer, $\mathcal{S}$ learns $c$ via simulation of $\mathcal{I}^n_{\mathsf{FCOT}}$.

Thus, the claim follows. □

## 7 SFE expansion for $t \leq (n-3)$

Our proof for SFE-expansion is structured in three lemmata. Each provides a limit on the maximum number of hybrid subfunctionalities that can be aborted. This implies a guarantee that some subfunctionalities cannot be aborted. Using this guarantee, we provide a protocol that uses $n$-party broadcast to expand $(n-1)$-party *global commitments* $\mathcal{I}^{n-1}_{\mathsf{COM}}$ to $n$-party global commitments $\mathcal{I}^n_{\mathsf{COM}}$. We then use this $n$-party commitment as a tool in the expansion of FCOT from $(n-1)$ parties to $n$ parties.

We perform our analysis in the Universal Composability Framework [Can01], which offers security guarantees regardless of the environment in which a protocol is executed. This allows for modular constructions using subfunctionalities.

**Lemma 15 (General subfunctionality abort).** *Let $n$ be the number of parties of which at most $0 \leq t < n$ are malicious. Let $\pi$ be an $\{\mathcal{I}^{n-1}, \mathcal{I}^n_{\mathsf{CG}}\}$-hybrid protocol that uses the Conflict Graph correctly; according to Lemma 10. If the adversary $\mathcal{A}$ aborts more than $t$ subfunctionality instances of cardinality $(n-1)$, then the Conflict Graph from $\mathcal{I}^n_{\mathsf{CG}}$ is biseparated.*

*Proof.* Denote the set of $n$ parties by $[P] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. Let $t' \leq t$ be the actual number of parties corrupted by the adversary $\mathcal{A}$. W.l.o.g., let $H = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_{n-t'}\}$ be the set of honest parties and let $C = \{\mathsf{P}_{n-t'+1}, \ldots, \mathsf{P}_n\}$ be the set of corrupted parties.

We now show that, if $\mathcal{A}$ aborts too many subfunctionalities, a set of parties must be separated from all others in the Complement Graph, thus leaving the corresponding Conflict Graph *biseparated*. Note that the Complement Graph is initially a complete graph and loses edges, when subfunctionalities are aborted. [4] Since honest parties are never in conflict, their mutual edges are never removed. We only consider subfunctionalities of cardinality $(n-1)$; thus, we have one subfunctionality instance for each excluded party $\mathsf{P} \in [P]$.

We call the set of corrupted parties that aborts a subfunctionality instance the *disruptor set* $D$. By $D_i$ we denote the disruptor set for the subfunctionality instance that excludes $\mathsf{P}_i$, and by $Z$ the set of corrupted parties that disrupted no subfunctionality so far. The disruptor sets corresponding to honest parties $D_1, \ldots, D_{n-t'}$ alongside with $Z$ partitions $C$, meaning that $D_1 \uplus \cdots \uplus D_{n-t'} \uplus Z = C$. $Z$ is disjoint with any $D_i$. If there was a nonempty intersection between two different disruptor sets $D_i$ and $D_j$, then this intersection $I_{ij} \coloneqq D_i \cap D_j$ would be in conflict with all parties who participated in $\mathcal{I}^{n-1}_{\mathsf{P}_i}$ and with the ones

[4] For simplicity, neglect the fact that edges could also be removed, if a party obviously deviates from the protocol.

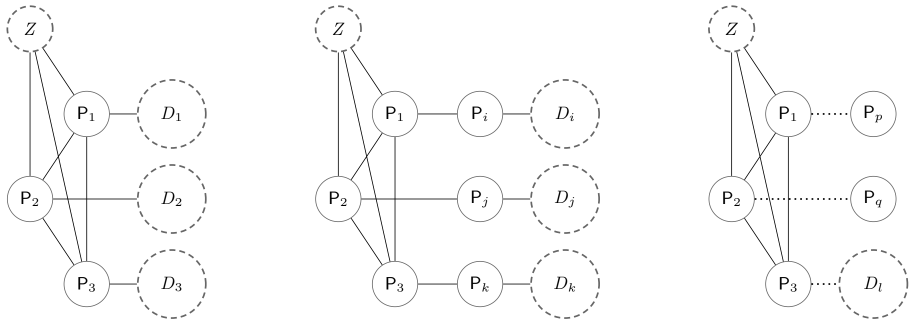


**Fig. 5:** Complementary Conflict Graph for the abort strategy with three honest parties Left: Result of aborting $\mathcal{I}_1^{n-1}$, $\mathcal{I}_2^{n-1}$ and $\mathcal{I}_3^{n-1}$. Middle: Same graph after additionally aborting $\mathcal{I}_i^{n-1}$, $\mathcal{I}_j^{n-1}$ and $\mathcal{I}_l^{n-1}$. Right: The $\mathsf{P}_3$ branch can never be terminated.

who participated in $\mathcal{I}_{\mathsf{P}_j}^{n-1}$, which is all parties. Thus, $I_{ij}$ would be completely separated in the Complement Graph. Consequently, the complementary CG loses all edges except the ones between $Z$ and $H$ and for each $i \in [n-t']$, the ones between $\mathsf{P}_i$ and $D_i$. The respective sets internally form a complete graph. See the left side of Fig. 5 for the case $t' = (n-3)$. Until now, only functionalities that omitted honest parties were aborted.

Recall that the complementary Conflict Graph must be connected in order for its complement graph not to be biseparated (Remark 2). Therefore, any subfunctionality $\mathcal{I}_{\mathsf{P}_i}^{n-1}$ for any omitted party $\mathsf{P}_i \in D_i \cup Z$ can only be aborted by other parties within the same set. Otherwise, it would be disconnected from its own set $D_i$ or $Z$, respectively. Since it already is disconnected from $H$, $Z$ and all other $D_j$, it would be completely isolated in the complementary Conflict Graph.

Consider, for example, the case where $D' \subsetneq D_1 \setminus \{\mathsf{P}_i\}$ disrupts $\mathcal{I}_{\mathsf{P}_i}^{n-1}$ for some $\mathsf{P}_i \in D_1$. This causes all remaining parties $\mathsf{P}_j \in D_1 \setminus \big(\{\mathsf{P}_i\} \cup D'\big)$ to declare a conflict with $D'$. Thus, those parties would be separated from $D'$. Since $\mathsf{P}_i$ was excluded from this subfunctionality, it remains connected to both $D_1$ *and* $D'$. This turns the Complement Graph into a *tree* containing subsets of parties as nodes, where $H$ is the root node containing all honest parties. The tree has one leaf that contains all non-disruptors $Z$ and $(n-t')$ branches, one for each honest party. Again, both $H$ and $Z$ internally form a complete graph. The abort of any $\mathcal{I}_{\mathsf{P}_j}^{n-1}$ for $\mathsf{P}_j \in D_i$ for an arbitrary $i \neq j$ thus leads to an extension of the respective branch. However, the adversary cannot abort the subfunctionality corresponding to the leaf node of each branch; there would have to be some party left that could abort the subfunctionality, which would then cause a conflict, thus separating a subset of the branch from the rest. As a consequence, at least $(n-t')$ subfunctionalities must succeed. The adversary can thus abort at most $t'$ subfunctionalities, before the Conflict Graph becomes biseparated. □

We consider two special cases: one where we have at least three honest parties ($t \leq n-3$) and one where the adversary can corrupt all but one parties ($t \leq n-1$).

In the former case, where $t \leq n-3$, only strictly less than $t$ subfunctionalities of cardinality $(n-1)$ can be aborted when using the Conflict Graph technique. For this case, Lemma 15 tightens to:

**Lemma 16 (Strong subfunctionality abort).** *Let $0 \leq t \leq (n-3)$. If $t$ or more subfunctionalities of cardinality $(n-1)$ are aborted, then the Conflict Graph is biseparated.*

*Proof.* It suffices to consider the case for $t$ aborted subfunctionalities. Assume that $t$ subfunctionalities are aborted. Each of the $t$ disruptors $\mathsf{D}'$ can have at most two edges in $\overline{G}$, namely the party $\mathsf{P}_i$ who was omitted in the functionality $\mathcal{I}^{n-1}_{\mathsf{P}_i}$ that $\mathsf{D}'$ aborted and the party $\mathsf{P}_j$ that disrupted $\mathcal{I}^{n-1}_{\mathsf{D}'}$. However, honest parties $\mathsf{P}_k$ can still have up to three neighbors in $\overline{G}$, namely two actual honest parties and one disruptor who aborted $\mathcal{I}^{n-1}_{\mathsf{P}_k}$. Honest parties can use this fact to determine its malicious neighbor: at least $(n-t-1)$ of its neighbors form a clique, while the malicious neighbor has only two neighbors of its own. Thus, the honest parties can declare a conflict with the party outside of their clique. This leaves the Conflict Graph such that all parties who are part of the clique in $\overline{G}$ form one partition, whereas all disruptors form the second partition. Thus, the Conflict Graph is biseparated. □

We now consider the case $t \leq (n-1)$. Here, Lemma 15 yields:

**Lemma 17 (Weak subfunctionality abort).** *Let $0 \leq t < n$. Either at most $(n-2)$ subfunctionalities of cardinality $(n-1)$ are aborted, or the Conflict Graph is biseparated.*

*Proof.* We only proof the case of a single honest party $\mathsf{P}$. If more parties are honest, less subfunctionalities can be aborted.

Denote the set of parties by $[P] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$; w.l.o.g., assume that $\mathsf{P} = \mathsf{P}_1$, meaning that $\mathsf{P}_1$ is honest. Furthermore, denote the corrupted parties as $C$.

After an abort of $\mathcal{I}^n_1$, the subgraph on $[P] \setminus \{\mathsf{P}_1\}$ must be biseparated. The only way this does not result in a biseparated Conflict Graph is if is all other parties are malicious. If the subgraph is additionally $t$-settled, the excluded party $\mathsf{P}_1$ can identify the settled set, causing a completely biseparated Conflict Graph. If the subgraph on $[P] \setminus \{\mathsf{P}_1\}$ only biseparated, we call the two partitions $S_0$ and $S_1$. We now focus our attention to the complement Conflict Graph, $\overline{G} = ([P], \overline{E})$. Note that biseparation of the Conflict Graph implies that no edge $e = (\mathsf{P}, \mathsf{P}') \in \overline{E}$ exists in $\overline{G}$ such that $\mathsf{P} \in S_0, \mathsf{P}' \in S_1$. However, in our Conflict Graph, $\mathsf{P}_1$ is still connected to both partitions. Each party in $S_0$ can only be aborted by a subset of $S_0$, since otherwise, a set of disruptors in $S_0$ of a functionality in $S_1$ would be in conflict with all other parties; the same argument holds for $S_1$. Every time a functionality omitting a party in any partition is aborted, that party is removed from the respective set, thereby shrinking the set. Hence, each set must have at least one party $\mathsf{P}_i$, whose hybrid functionality $\mathcal{I}^n_{\mathsf{P}_i}$ cannot be aborted, without creating a biseparated Conflict Graph. $\square$

We now have two limits on the maximal number of hybrid functionalities that the adversary can abort before causing a biseparated Conflict Graph. Using them, we are able to expand the (SFE-incomplete) functionality Global Commitment (GCOM) from $\mathcal{I}^{n-1}_{\mathsf{COM}}$ to $\mathcal{I}^n_{\mathsf{COM}}$.

**Lemma 18 (COM expansion).** *Let $n$ be the number of parties of which at most $0 \le t \le (n-2)$ are malicious; subject to $n \in \mathcal{O}(\ln \lambda) \vee n-t \in \mathcal{O}(1)$. There exists a protocol $\pi$ in the $\{\mathcal{I}^{n-1}_{\mathsf{COM}}, \mathcal{I}^n_{\mathsf{BC}}\}$-hybrid model that securely UC-realizes $\mathcal{I}^n_{\mathsf{COM}}$:*

$$\{\mathcal{I}^{n-1}_{\mathsf{COM}}, \mathcal{I}^n_{\mathsf{BC}}\} \rightsquigarrow \mathcal{I}^n_{\mathsf{COM}} \tag{5}$$

*Proof.* We only consider bit-commitments, which can be canonically extended to string-commitments. Let the set of parties be $[P] = \{\mathsf{C}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$ and let $b$ be the bit that the committer $\mathsf{C}$ commits to. We present a protocol that uses $(n-1)$ COM-subfunctionality instances $\mathcal{I}^{n-1}_{R_l}$ for each excluded receiver $\mathsf{R}_l$. Additionally, it uses the Conflict Graph $\mathcal{I}^n_{\mathsf{CG}}$ which can be realized using $\mathcal{I}^n_{\mathsf{BC}}$; here the condition $n \in \mathcal{O}(\ln \lambda) \vee n-t \in \mathcal{O}(1)$ comes into play again. For the sake of simplicity, we denote by $\mathcal{I}^{n-1}_{\mathsf{P}_i}$ the Commitment-subfunctionality which omits $\mathsf{R}_i$. Denote $\mathsf{C}$'s input to $\mathcal{I}^{n-1}_l$ by $b_l$ and the abort set for $\mathcal{I}^{n-1}_l$ by $D_l$. Let $I_0$ be the set of subfunctionalities where $b_l = 0$, let $I_1$ be the set of subfunctionalities where $b_l = 1$, and let $I_\times$ be the set of subfunctionalities that have previously been aborted. Note that $\mathcal{I}^n_{\mathsf{CG}}$ ensures that $I_\times$ is public knowledge; the abort of a subfunctionality $\mathcal{I}^{n-1}_l$ results in a biseparated subgraph on $[P] \setminus \{\mathsf{R}_l\}$ and if a subgraph is biseparated, the corresponding subfunctionality is considered aborted. It holds that $|I_0| + |I_1| + |I_\times| = (n-1)$, since there are $(n-1)$ instances of $\mathcal{I}^{n-1}_{\mathsf{P}_i}$.

In the following we give a description of the protocol.

Let $b$ be the bit $\mathsf{C}$ wants to commit to. For runtime restrictions, we assume that the unary security parameter $1^\lambda$ is also input.

**Protocol:**

1. On input $(\texttt{commit}, b)$ with $b \in \{0,1\}$ from $\mathcal{Z}$ to $\mathsf{C}$, $\mathsf{C}$ sends $(\texttt{commit}, b)$ to $\mathcal{I}^{n-1}_l$ for all $l \in \{1, \ldots, n-1\}$.
2. Once $\mathsf{R}_l$ has received output from any $\mathcal{I}^{n-1}_{\mathsf{P}_j}$ for $j \neq l$, the receiver $\mathsf{R}_l$ checks $|I_\times|$:

   **If $t \le n-3$ and $|I_\times| \ge t$,** then the Conflict Graph is biseparated due to Lemma 16. $\mathsf{R}_l$ aborts with the identified set of parties.

   **If $|I_\times| \ge n-1$,** then the Conflict Graph is biseparated due to Lemma 17. $\mathsf{R}_l$ aborts with the identified set of parties.

   **Otherwise,** $\mathsf{R}_l$ outputs $(\texttt{receipt commit})$.
3. On input $(\texttt{unveil})$ from $\mathcal{Z}$ to $\mathsf{C}$, $\mathsf{C}$ sends $(\texttt{unveil})$ to $\mathcal{I}^{n-1}_l$ for all $l \in \{1, \ldots, n-1\}$.
4. Once $\mathsf{R}_l$ has received output from all $\mathcal{I}^{n-1}_{\mathsf{P}_j}$ for $j \neq l$, $\mathsf{R}_l$ sends $(\texttt{receipt}, \mathsf{R}_l)$ to $\mathcal{I}^n_{\mathsf{BC}}$.

5. Once $\mathsf{R}_l$ has received $(\texttt{receipt}, \mathsf{R}_j)$ from all $\mathsf{R}_j$ for $j \neq l$ via $\mathcal{I}^n_{\mathsf{BC}}$, $\mathsf{R}_l$ checks $|I_\times|$.

   **If** $|I_\times| \geq n-1$**,** then the Conflict Graph is biseparated due to Lemma 17. $\mathsf{R}_l$ aborts with the identified set of parties.

   **If** $|I_\times| = n-2$**,** *excluded* receiver $\mathsf{R}_l$, i.e. $\mathcal{I}^{n-1}_l$ has been aborted, concurs with the output of their (only) neighbor $\mathsf{P}$ in $\overline{G}$. $\mathsf{R}_l$ does so by broadcasting $(\texttt{help})$ whereupon $\mathsf{P}$ broadcasts its output. If $\mathsf{P}$ is the committer, the committer broadcasts its bit $b$. Otherwise, $\mathsf{P}$ is a receiver who checks all of its unveiled bits for consistency. If all but one bit are equal, then this bit $b$ is the output $(\texttt{unveil}, b)$. The receiver broadcasts $(\texttt{unveil}, b)$ to signal to $\mathsf{R}_l$ to output the same. If inconsistent bits have been received, then $\mathsf{P}$ broadcasts $(\texttt{abort}, \mathsf{C})$ and outputs the same.

   **If** $|I_\times| = n-3$**,** essentially the same strategy as in the previous case applies. Only, here are up to three honest parties possible, therefore excluded receivers only concur with neighbors in $\overline{G}$ which have at least three neighbors themselves. These neighbors are naturally included in all subfunctionalities that have not been aborted.

   **If** $|I_\times| \leq n-4$**,** $\mathsf{R}_l$ checks the unveil messages:

   Since $|I_\times| \leq n-4$ each receiver gets at least three messages; there is also $\mathcal{I}^{n-1}_0$, which omits the Committer $\mathsf{C}$, which is not used. If all but one unveilings are consistent with $b'$, $\mathsf{R}_l$ outputs $(\texttt{unveil}, b')$.

   Otherwise, $\mathsf{R}_l$ outputs $(\perp)$ and sends $\{\texttt{conflict}, \mathsf{C}\}$ to $\mathcal{I}^n_{\mathsf{CG}}$.

   In our synchronous model, messages sent by a Sender are guaranteed to be received by the dedicated Receiver. If $\mathsf{R}_l$ doesn't obtain a receipt from all other parties in Step 5, then at least one of the following has to be true: (a) the Conflict Graph is already biseparated after Step 2, which implies that all honest parties have aborted; or (b) the missing message's sender $\mathsf{R}_j$ must be corrupted. Hence a missing message from $\mathsf{R}_j$ is considered an abort of $\mathcal{I}^n_{\mathsf{BC}}$ from $\mathsf{R}_j$. Since this semisettles the entire Conflict Graph, the $\mathcal{I}^n_{\mathsf{COM}}$ functionality is aborted.

The intuition behind the protocol is the following: The committer $\mathsf{C}$ commits its bit $b$ to all subfunctionalities, which gives honest receivers consistent opening information. The adversary has two levers to disturb the protocol: One is to abort subfunctionalities, thus increasing $|I_\times|$; we have shown in Step 16, that this is only possible for up to $t$ aborts, before the Conflict Graph becomes biseparated. The other option the adversary has is to let a corrupted committer use different bits in different subfunctionalities. If at least four subfunctionalities are not aborted, then the second adversarial strategy no longer works, since all receivers will notice a sufficient inconsistency in the subfunctionalities. Therefore the adversary has to abort many subfunctionalities. This increases the honest parties knowledge about the identity of malicious parties sufficiently for honest parties to identify each other. Finally, if too many subfunctionality are aborted, such that only one is left, the Conflict Graph has sufficiently many edges that a identification is possible. If a receiver does not accept, one of two cases must have happened: Both in Step 2 and Step 5, $\mathsf{R}$ rejects if $|I_\times| \geq t$. In that case, it follows from Lemma 16 and Lemma 17 that the resulting Conflict Graph is biseparated, hence identification for an abort is possible. If $t \leq (n-3)$ Lemma 16 applies directly, whereas if $t \geq (n-2)$ and $|I_\times| > t$ Lemma 17 applies.

We note that our strategy does not work directly for $t \leq (n-1)$ because the honest party has no other honest party to rely on, if too many subfunctionalities are aborted.

Proving security of this protocol is straightforward. First note that in every case the behavior of honest receivers in unambiguous. We can thus formulate a coherent simulator. In particular, the dummy adversary's and the corrupted parties' local in- and output is forwarded from and to the environment.

**On output** (`receipt commit`) from $\mathcal{I}^n_{\mathsf{COM}}$, the simulator gives local input to all uncorrupted simulated parties as follows: The simulator gives the uncorrupted $\mathsf{C}$ local input $(\texttt{commit}, 0)$, hence $\mathsf{C}$ inputs $(\texttt{commit}, 0)$ into all $\mathcal{I}^{n-1}_l$.

**On output** $(\texttt{unveil}, m)$ from $\mathcal{I}^n_{\mathsf{COM}}$, the simulator lets the simulated functionalities $\mathcal{I}^{n-1}_l$ output $(\texttt{unveil}, m)$. This is possible because $\mathcal{I}^{n-1}_l$ is a simulated functionality and thus fully under the simulator's control.

**Once** all broadcasts $(\texttt{receipt}, \mathsf{R}_l)$ from $\mathcal{I}^n_{\mathsf{BC}}$ have been received, the simulator lets $\mathsf{C}$ input (`unveil`) into the ideal functionality $\mathcal{I}^n_{\mathsf{COM}}$.

**On input** $\big(\texttt{abort}, C'\big)$ from $\mathcal{Z}$ for $\mathcal{I}^{n-1}_l$, the simulator sends $\big(\texttt{abort}, C'\big)$ to $\mathcal{I}^{n-1}_l$.

**If** the (simulated) Conflict Graph becomes biseparated, the simulator aborts $\mathcal{I}^n_{\mathsf{COM}}$ with the malicious partition $\big(\texttt{abort}, C'\big)$.

Finally, we describe the simulator's behavior when handling messages from/to malicious parties.

**On input** $(\texttt{commit}, m_l)$ from corrupted $\mathsf{C}$ to $\mathcal{I}^{n-1}_l$, the simulator lets $\mathsf{C}$ pass $(\texttt{commit}, m_l)$ on to $\mathcal{I}^{n-1}_l$. After receiving local input for all (non-aborted) subfunctionalities, the simulator lets $\mathsf{C}$ input $(\texttt{commit}, m)$ into $\mathcal{I}^n_{\mathsf{COM}}$ where $m$ is the majority of all $m_l$. If there is no majority then $\mathsf{C}$ inputs a random bit $m$. The inputs $(\texttt{commit}, m_l)$ are naturally passed to the simulated $\mathcal{I}^{n-1}_l$.

**On input** (`unveil`) from corrupted $\mathsf{C}$ to $\mathcal{I}^{n-1}_l$, the simulator lets $\mathsf{C}$ forward (`unveil`) to $\mathcal{I}^{n-1}_l$.

**On input** (`receipt commit`) from corrupted $\mathsf{R}_l$ to $\mathcal{I}^{n-1}_{\mathsf{P}_j}$, the simulator forwards (`receipt commit`) in the name of $\mathsf{R}_l$.

The key to a coherent simulation is that the protocol ensures that the outputs of all (honest) receivers are consistent, this can be leveraged by the simulator to abort the functionality on invalid inputs of the (corrupted) committer. $\square$

To provide SFE-expansion as sketched in Fig. 3, the one missing step required is an expansion of FCOT. Denote the parties as sender $\mathsf{S}$, receiver $\mathsf{R}$ and witnesses $\mathsf{W}_1$ through $\mathsf{W}_{n-2}$. We call *type-1* subfunctionalities $\mathcal{I}^{n-1}_{\mathsf{P}_i}$ for all $i \in \{1, \ldots, n-2\}$ which exclude a witness $\mathsf{W}_i$ and *type-2* subfunctionalities both $\mathcal{I}^{n-1}_{n-1}$ and $\mathcal{I}^{n-1}_n$, which exclude the Sender $\mathsf{S}$ and the Receiver $\mathsf{R}$, respectively. In abuse of notation, we refer to them as $\mathcal{I}^{n-1}_{\mathsf{S}}$ and $\mathcal{I}^{n-1}_{\mathsf{R}}$.

**Corollary 2.** *Let $\pi$ be a protocol that securely realizes $\mathcal{I}^n_{\mathsf{FCOT}}$. At least two type-1 functionalities cannot be aborted, otherwise the Conflict Graph is biseparated.*

This follows from Lemma 16 with $t = (n-3)$, which states that only strictly less than $(n-3)$ subfunctionalities of cardinality $(n-1)$ can be aborted without producing a biseparated Conflict Graph. Conversely, at least four subfunctionalities of cardinality $(n-1)$ must succeed, two of which may be $\mathcal{I}_{\mathsf{S}}^{n-1}$ and $\mathcal{I}_{\mathsf{R}}^{n-1}$. Hence the remaining two must be of type-1.

Now, we use Corollary 2 to construct a protocol that securely realizes $\mathcal{I}_{\mathsf{FCOT}}^{n}$ from $\mathcal{I}_{\mathsf{FCOT}}^{n-1}$ and $\mathcal{I}_{\mathsf{COM}}^{n}$.

**Lemma 19 (FCOT expansion).** *Let $n$ be the number of parties of which at most $0 \leq t \leq (n-3)$ are malicious; subject to $n \in \mathcal{O}(\ln \lambda) \vee n - t \in \mathcal{O}(1)$. There is a protocol $\pi_{\mathsf{FCOT}}^{n}$ that UC-realizes $\mathcal{I}_{\mathsf{FCOT}}^{n}$ in the $\left\{\mathcal{I}_{\mathsf{FCOT}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}\right\}$-hybrid model:*

$$\left\{\mathcal{I}_{\mathsf{FCOT}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}\right\} \rightsquigarrow \mathcal{I}_{\mathsf{FCOT}}^{n} \tag{6}$$

*Proof.* We describe the protocol $\pi_{\mathsf{FCOT}}^{n}$. Denote the parties as sender $\mathsf{S}$, receiver $\mathsf{R}$ and witnesses $\mathsf{W}_1$ through $\mathsf{W}_{n-2}$.
The protocol is iteration-based. Let $I_\times$ be the set of type-1 subfunctionalities that have been aborted. There are $T \coloneqq (n - 2 - |I_\times|)$ type-1 subfunctionalities that have **not** yet been aborted. In each iteration of the protocol enumerate these as $\left(\mathcal{I}_1^{n-1}, ..., \mathcal{I}_T^{n-1}\right)$. The protocol uses $r \in \omega(n \ln \lambda)$ sessions of each remaining subfunctionality $\mathcal{I}_l^{n-1} \notin I_\times$. We implicitly use the string variant of the commitment functionality, which can be constructed from the aforementioned bit commitment. We consider a subfunctionality to be aborted, if the Conflict Graph of its participants is biseparated. Note that $\mathcal{I}_{\mathsf{BC}}^{n}$ follows trivially from a $\mathcal{I}_{\mathsf{COM}}^{n}$ by immediately unveiling the commitment. Furthermore, we showed in Lemma 18, that $\mathcal{I}_{\mathsf{COM}}^{n}$ follows from $\mathcal{I}_{\mathsf{COM}}^{n-1}$. We show in Section 6 that $\mathcal{I}_{\mathsf{COM}}^{n-1}$ follows from $\mathcal{I}_{\mathsf{FCOT}}^{n-1}$. Hence, our construction works in the $\left\{\mathcal{I}_{\mathsf{COT}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}\right\}$-hybrid model; we still use the terms $\mathcal{I}_{\mathsf{COM}}^{n-1}$ and $\mathcal{I}_{\mathsf{CG}}^{n}$ in our proofs.

The central idea of the protocol is to use secret sharings of the sender's messages to perform multiple FCOTs of cardinality $(n-1)$ and to globally commit to these shares. We use a *cut-and-choose* trick, where some shares are unveiled to ensure that the FCOTs and Commitments are equal. After the Oblivious Transfer, the receiver commits to the received shares. The secret sharing implies that he cannot unveil a different choice bit afterwards. We assume inputs $\mathsf{S}(m_0, m_1)$, $\mathsf{R}(c)$ and $\mathsf{W}_l(\varepsilon)$ together with the implicit unary security parameter $1^\lambda$.

The protocol is parameterized by two natural numbers $T \in \text{poly}(\lambda)$ and $r \in \text{poly}(\lambda)$, which define the amount of distributed additive shares.

**Protocol iteration:**

1. If the Conflict Graph becomes biseparated, all honest parties abort with their opposite partition.
2. On input $(\texttt{messages}, m_0, m_1)$ with $m_0, m_1 \in \{0,1\}$ from $\mathcal{Z}$ to $\mathsf{S}$, $\mathsf{S}$ creates an additive $T$-sharing of $m_0$ and $m_1$ called $(\mu_j^0)_{j=1..T}$ and $(\mu_j^1)_{j=1..T}$, respectively. $\mathsf{S}$ then creates an $r$-sharing $(\alpha_{j,i}^b)_{i=1..r}$ for each share $\mu_j^b$ with a $(2r/3)$-threshold secret sharing scheme. $\mathsf{S}$ inputs $(\texttt{messages}, \alpha_{j,i}^0, \alpha_{j,i}^1)$ into the $i$-th session of $\mathcal{I}_{\mathsf{P}_j}^{n-1}$ for all $i \in [r]$ and $j \in [T]$ and commits globally to each share by sending $\alpha_{j,i}^0$ resp. $\alpha_{j,i}^0$ into $\mathcal{I}_{\mathsf{COM}}^{n}$ for each $i \in [r]$ and $j \in [T]$.

3. On input $(\texttt{choice}, c)$ with $c \in \{0,1\}$ from $\mathcal{Z}$ to $\mathsf{R}$, $\mathsf{R}$ sends $(\texttt{choice}, c)$ to all remaining subfunctionalities $\mathcal{I}_l^{n-1}$ for all $l \in [T]$. Additionally, $\mathsf{R}$ sends $c$ to $\mathcal{I}^n_{\mathsf{COM}}$.
4. If subfunctionality $\mathcal{I}_l^{n-1} \notin I_\times$ is aborted, the current iteration ends and $\mathcal{I}_l^{n-1}$ is added to $I_\times$.
5. When a party $\mathsf{P}$ has received $(\texttt{receipt messages})$ resp. $(\texttt{receipt choice})$ from all $r$ sessions of all type-1 FCOT-subfunctionalities that include $\mathsf{P}$, $\mathsf{P}$ outputs $(\texttt{receipt messages})$ resp. $(\texttt{receipt choice})$.
6. When no subfunctionality has been aborted in this iteration and all $T$ subfunctionalities have successfully finished their OT-phase with output $(\texttt{receipt transfer}, \perp)$ to $\mathcal{S}$, $\mathsf{R}$ learns $m_c$: it obtains all $r$ shares $\alpha^c_{j,i}$ of all $T$ additive shares $\mu^c_j$ of $m_c$. Each $\mathsf{W}_i$ and $\mathsf{R}$ verify the integrity of the sender's commitments, by verifying that the sender's global commitments indeed contain the correct input used for the FCOT-subfunctionalities. Each party $\mathsf{P} \in [P] \setminus \{\mathsf{S}\}$ broadcasts the set of $r/10n$ indices $i \in [r]$ per subfunctionality $l \in [T]$. For each index $(i,l)$, the sender has to unveil the corresponding global commitment and the FCOT-subfunctionality.
7. Upon receiving $(\texttt{unveil message}, b)$ from $\mathcal{Z}$ to $\mathsf{S}$, $\mathsf{S}$ sends $(\texttt{unveil})$ to all $\mathcal{I}^n_{\mathsf{COM}}$. Upon receiving their shares, the parties output $(\texttt{unveil message}, b, m_b)$.
8. Upon receiving $(\texttt{unveil choice})$ from $\mathcal{Z}$ to $\mathsf{R}$, $\mathsf{R}$ inputs $(\texttt{unveil choice})$ into all FCOT-functionalities and $(\texttt{unveil})$ into $\mathcal{I}^n_{\mathsf{COM}}$ to unveil $c$. The other parties first check consistency of all unveiled choice bits. If not consistent, all honest parties immediately abort with $(\texttt{abort}, \mathsf{R})$. If the choice bits are consistent $c'$, then the other parties check consistency between the unveiled choice bits of the FCOT-sessions and the global commitment. If internally inconsistent or inconsistent with the global commitment, the affected FCOT is considered aborted, since all participating party are able to identify the receiver as malicious.

Now, we give a description of the simulator.

**On input** $(\texttt{messages}, \alpha^0_{j,i}, \alpha^1_{j,i})$ from corrupted $\mathsf{S}$ to the $i$-th session of $\mathcal{I}^{n-1}_{\mathsf{P}_j}$, the simulator lets $\mathsf{S}$ forward the input to the simulated $\mathcal{I}^{n-1}_{\mathsf{P}_j}$.

**On input** $(\texttt{messages}, \alpha^0_{j,i}, \alpha^1_{j,i})$ from corrupted $\mathsf{S}$ to all sessions of all remaining FCOT-functionalities, the simulator computes $m_0$ and $m_1$ from the obtained shares. Then the simulator lets $\mathsf{S}$ input $(\texttt{messages}, m_0, m_1)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$.

**On input** $(\texttt{commit}, \alpha^b_{j,i})$ from corrupted $\mathsf{S}$ to $\mathcal{I}^n_{\mathsf{COM}}$, the simulator lets $\mathsf{S}$ input into the simulated $(\texttt{commit}, (\alpha^b_{j,i}, b, i, j))$ to $\mathcal{I}^n_{\mathsf{COM}}$.

**On input** $(\texttt{choice}, c_{j,i})$ from corrupted $\mathsf{R}$ for the $i$-th session of $\mathcal{I}^{n-1}_{\mathsf{P}_j}$, the simulator lets $\mathsf{S}$ forward the input to the simulated $\mathcal{I}^{n-1}_{\mathsf{P}_j}$.

**On input** $(\texttt{choice}, c_{j,i})$ from corrupted $\mathsf{R}$ to all $i$-th sessions of all remaining FCOT-functionalities and $(\texttt{commit}, c)$ for $\mathcal{I}^n_{\mathsf{COM}}$, the simulator lets $\mathsf{R}$ input $(\texttt{choice}, c)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$.

**On output** $(\texttt{receipt transfer}, \perp)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$, the simulator gives local input to simulated parties as follows:
If the receiver is malicious, then $\mathcal{S}$ learns $m_c$ from the ideal $\mathcal{I}^n_{\mathsf{FCOT}}$ in the

name of $\mathsf{R}$ through $(\texttt{receipt transfer}, m_c)$. Otherwise, the simulator gives the uncorrupted $\mathsf{R}$ local input $(\texttt{choice}, 0)$. If the sender is uncorrupted, the simulator gives $\mathsf{S}$ local input $(\texttt{messages}, m_0, m_1)$ with $m_{1-c} = 0$. If the receiver is uncorrupted, the simulator also uses $m_c = 0$. Then, the simulator simulates the protocol program with the respective inputs. If any party is malicious, its inputs are directly forwarded to the simulated hybrid functionalities. Consequently, the simulated dummy adversary receives $(\texttt{receipt transfer}, \perp)$ from each simulated FCOT-subfunctionality which is forwarded to the environment.

**On input** $(\texttt{unveil messages}, b)$ from corrupted $\mathsf{S}$ to all sessions of all remaining FCOT-functionalities, the simulator lets $\mathsf{S}$ input $(\texttt{unveil messages}, b)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$.

**On output** $(\texttt{unveil message}, b, m_b)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$, the simulator unveils all shares of $m_b$ from all FCOT-subfunctionalities and all their commitments. If the local input $(\texttt{messages}, m_0, m_1)$ of the simulated sender matches with the unveiled $m_b$, then the simulation is valid.

However, if the simulated sender's input is not equal to the ideal uncorrupted sender's input, the simulator must equivoke some of the FCOTs and COMs. Therefore, the simulator fabricates additive shares $\mu^b_j$ and sub-shares $\alpha^b_{j,i}$ that encode $m_b$ but still are consistent with the shares that the environment learned so far. The simulator must be able to equivoke more than $r/3$ (out of $r$) shares of a single FCOT-subfunctionality. This is possible because the consistency check in the OT-phase only leaks less than $r/10$ per subfunctionality to the environment, leaving more than $9r/10$ for the simulator to equivoke.

If the receiver is malicious, then the environment also learns the shares that the receiver obtains during the OT-phase. However, if the receiver is malicious, then the simulator already learned $m_c$ prior to giving the simulated sender its input, hence the simulated sender's input $m_c$ is consistent with $m_b$ if $b = c$. Otherwise the environment only has less than $r/2$ shares per FCOT, leaving $r - (r/2 + r/10) > r/3$ for the simulator to equivoke.

Furthermore, the simulator knows exactly which shares can be equivoked since every share that the environment obtains comes from the simulator.

**On input** $(\texttt{unveil choice})$ from a corrupted $\mathsf{R}$ to all remaining FCOT-subfunctionalities, the simulator lets $\mathsf{R}$ input $(\texttt{unveil choice})$ into $\mathcal{I}^n_{\mathsf{FCOT}}$.

**On output** $(\texttt{unveil choice}, c)$ from $\mathcal{I}^n_{\mathsf{FCOT}}$, the simulator lets all sessions of all simulated FCOT-subfunctionalities $\mathcal{I}^{n-1}_l$ output $(\texttt{unveil choice}, c)$ to all parties and the dummy adversary.

**On input** $\big(\texttt{abort}, C'\big)$ for $\mathcal{I}^{n-1}_l$, the simulator aborts $\mathcal{I}^{n-1}_l$ with $\big(\texttt{abort}, C'\big)$.

**If** the (simulated) Conflict Graph becomes biseparated, the simulator aborts $\mathcal{I}^n_{\mathsf{COM}}$ with the malicious partition $\big(\texttt{abort}, C'\big)$.

Also, if an adversarial sender wanted to prepare its share commitments in order to equivoke its message afterwards, then it would have to alter more than $r/3$ shares. Recall that $r \in \omega(n \ln \lambda)$. Because at least $r/10n \in \omega(\ln \lambda)$ shares are uniformly randomly opened for control, the probability that the sender chooses more than $r/3$ shares none of which are controlled is negligible. More formally, the probability that none of $r/3$ altered shares are altered is controlled is bounded by

$$\begin{aligned} p &= \prod_{i=0}^{r/10n} \frac{r}{3}\frac{1}{r-i} \\ &\leq \prod_{i=0}^{r/10n} \frac{r}{3}\frac{1}{r-r/10n} \\ &= 3^{r/10n+1}(1-1/10n)^{r(1-1/10n)} \\ &\leq 3e^{r/n\cdot\ln(3)/10-r} \\ &= 3e^{-r(1-\ln(3)/10n)} \\ &\leq 3^{11/10}e^{-r} \end{aligned} \tag{7}$$

which is negligible by the the bound on $r \in \omega(n \ln \lambda)$. □

Intuitively, security follows from the fact that each FCOT-subfunctionality unveils less than $r/10$ shares, but reconstruction requires $2r/3$ shares. A user can learn at most $11r/10$ shares, whereas $4r/3$ would be required to learn both messages. If $\mathsf{S}$ tries to unveil a different value than its original input, it would have to deviate in more than $r/3$ shares of any one subfunctionality. For each subfunctionality, a party can probe $r/10n$ shares. Thus, $\mathsf{S}$ has negligible probability of successfully deviating from the original FCOT-input in the required number $r/3$ of global commitments.

The integrity check in the OT-phase ensure that the values must match the FCOT-inputs with overwhelming probability. From Corollary 2, it follows that at least for two $\mathcal{I}_l^{n-1} \notin I_\times$, all $r$ sessions must be unveiled. Also, $\mathsf{R}$ unveils the commitments to its received shares. If $\mathsf{R}$ tries to learn both messages, $\mathsf{R}$ to have input $(1-c)$ into all sessions of at least two FCOT-subfunctionalities; in that case, $\mathsf{R}$ cannot learn $m_c$, since $\left(\mu_j^c\right)_{j\in\{T\}}$ is an additive sharing and thus requires all shares for reconstruction.

**Theorem 8 (SFE expansion).** *Let $n$ be the number of parties of which at most $0 \leq t \leq (n-3)$ are malicious; subject to $n \in \mathcal{O}(\ln\lambda) \vee n-t \in \mathcal{O}(1)$. The functionality $\mathcal{I}_{\mathsf{SFE}}^n$ can be UC-realized in the $\left\{\mathcal{I}_{\mathsf{SFE}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^n\right\}$-hybrid model:*

$$\left\{\mathcal{I}_{\mathsf{SFE}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^n\right\} \rightsquigarrow \mathcal{I}_{\mathsf{SFE}}^n \tag{8}$$

*Proof.* This theorem follows from combining our previous lemmas according to Fig. 3. □

If we additionally assume a *multicast* functionality were the recipient set can be chosen by the sender, then we can tighten our result for any $t \leq n-3$. By induction we can deduce an upper bound of the minimal complete cardinality for each number of maximal corruptions; as we have shown $\mathcal{I}_{\mathsf{SFE}}^{n-2}$ and $\mathcal{I}_{\mathsf{BC}}^n$ realize $\mathcal{I}_{\mathsf{SFE}}^{n-1}$ for $t \leq n-3$. But this also holds for $t \leq n-4$, then we have that $\mathcal{I}_{\mathsf{SFE}}^{n-3}$ and

$\mathcal{I}_{\mathsf{BC}}^{n-1}$ realizes $\mathcal{I}_{\mathsf{SFE}}^{n-2}$. By induction we get $\mathcal{I}_{\mathsf{SFE}}^{t+2}$ and all $\mathcal{I}_{\mathsf{BC}}^{i}$ for $i \in \{t+3, \ldots, n\}$ realize $\mathcal{I}_{\mathsf{SFE}}^{n}$.

However, when the number of inductions is linear in $n$ then a slightly lower bound $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$ applies regardless of the hardness of computing $X$. This restriction comes from the polynomial runtime in $n$ of the SFE-expansion. For $n/2$ inductions we obtain an overall runtime of at most

$$(n)^c \cdot (n-1)^c \cdots (n/2)^c = (n!/(n/2)!)^c \tag{9}$$

for some constant $c$ which is polynomial in $\lambda$ iff $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$. For a better bound on $n$ one would need to give an SFE-protocol with lower runtime, e.g. polylogarithmic runtime in $n$ would support arbitrary $n \in \text{poly}(\lambda)$.

**Corollary 3.** *For $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$ and up to $t$ corruptions, it holds for the minimal complete cardinality that $k^*(t) \leq t+2$ (given broadcast).*

It is an interesting insight that composing functionalities is possible when three parties are honest. Intuitively, this is the case because in each subfunctionality, we now have a guarantee that *at least* two parties are honest such that upon abort they share the same information about the disruptors.

## 8 Summary and Outlook

*Summary.* We introduced a new tool for the analysis of protocols in the framework of Identifiable Abort [IOZ14]: the Conflict Graph (CG). We tightly linked properties of protocols with Identifiable Abort to verifiable graph properties of the Conflict Graph, namely biseparation, which we hope will contribute to further research in the field of Identifiable Abort. In particular, we show that the biseparation of the CG of any protocol is both necessary and sufficient for an Identifiable Abort.

We introduced a new variant of Oblivious Transfer, namely Fully Committed Oblivious Transfer, which we show to be *as powerful* as Secure Function Evaluation in the setting of IA, yet which is easier to analyze.

Using the Conflict Graph and the SFE-completeness of FCOT we constructed $n$-party SFE from $(n-1)$-party SFE and a broadcast channel. Our construction holds even against statistical adversaries, as long as at least three parties are honest. This SFE-expansion yields an upper bound on the minimal complete cardinality in the sense of [FGM+01]: $k^*(t) \leq t+2$ for $n \in \mathcal{O}(\ln\lambda/\ln\ln\lambda)$ given broadcast. This bound complements the results of [RB89; Bea90] in the dishonest-majority setting; compare Fig. 1:

In conclusion our CG-technique provides a new way of furthering the understanding of IA.

*Outlook.* Since we introduce a new methodology to analyze protocols Identifiable Abort there are naturally many open questions. The first open problem is to improve the efficiency of our protocols or to remove the broadcast requirement.

Another open issue is to tighten our upper bound on $k^*$, here one would need to provide abort-lemmas similar to Lemmas 15 to 17 for subfunctionalities of cardinality $n-2$ or less.

A third research direction is to clarify the time complexity of computing the settled set $X$ as defined in Definition 2. Either it is efficiently computable for all $t$, which could be useful for efficient broadcast protocols in the style of [CFF$^+$05], or deducing a conflict graph is actually harder for smaller $t$ than for larger $t$. Lastly, one could look for another algorithm to compute $G^*$ than DeduceCG that is efficiently computable without needing to compute $X$.

# Acronyms

**AA** Anonymous Abort
**BC** Broadcast
**CG** Conflict Graph
**COM** Commitment
**COT** Committed Oblivious Transfer
**CRS** Common Reference String
**DLog** Discrete Logarithm
**DoS** Denial-of-Service
**FCOT** Fully Committed Oblivious Transfer
**GCOM** Global Commitment
**GOD** Guaranteed Output Delivery
**IA** Identifiable Abort
**IF** Integer Factorization
**LWE** Learning with Errors
**MPC** Multi-Party Computation
**MVC** Minimal Vertex Cover
**OT** Oblivious Transfer
**SFE** Secure Function Evaluation
**UC** Universal Composability
**VC** Vertex Cover

# Symbols

**adversary** real/hybrid adversary
sub-share of a message $m$
**complementary edges** complement of the conflict edges
**complement graph** complement of the conflict graph
**committer** committing party
**corrupted parties** set of all corrupted parties
**disruptor party** party that maliciously deviates from the protocol
**disruptors** set of disruptor parties
**conflict edges** contains a pair of parties $\{\mathsf{P}, \mathsf{P}'\}$ if $\mathsf{P}$ broadcasts $(\texttt{conflict}, \mathsf{P}*)$
**partition** partition of a graph
empty string
ideal functionality with unspecified output property
**conflict graph** represents all public conflicts between parties
additive share of a choice bit $c$
**honest parties** set of all honest parties
set of aborted subfunctionalities
ideal functionality with identifiable abort
**complete minimal cardinality** minimal cardinality of a functionality that is complete for $n$-party Multi-Party Computation (MPC) for a given $t$
**MVCs** set of MVCs
additive share of a message $m$
(total) number of parties
**negligible functions** set of negligible functions with respect to a given argument
**non-disruptors** set of corrupted parties that never disrupt
choice index of a share
**overwhelming functions** set of overwhelming functions w.r.t. an argument
**parties** set of all parties
**party** protocol party
combined protocol of the IPS-compiler
outer protocol of the IPS-compiler
**receiver** receiving party
inner protocol of the IPS-compiler
**selector function** specified by a protocol
**sender** sending party
**simulator** simulates a protocol execution using a ideal functionality
maximal number of corrupted parties
**transcript**
**witness** passive party that only receives a receipt
set of parties that can be identified as malicious by an outsider
**environment**

## References

[BCG93] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In pages 52–61, 1993.

[Bea90] D. Beaver. Multiparty protocols tolerating half faulty processors. In pages 560–572, 1990.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In pages 1–10, 1988.

[Bon98] D. Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423, 1998. Invited paper.

[BOS16] C. Baum, E. Orsini, and P. Scholl. Efficient secure multiparty computation with identifiable abort. In pages 461–490, 2016.

[Can00] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In pages 136–145, 2001.

[CF01] R. Canetti and M. Fischlin. Universally composable commitments. In pages 19–40, 2001.

[CFF$^+$05] J. Considine, M. Fitzi, M. K. Franklin, L. A. Levin, U. M. Maurer, and D. Metcalf. Byzantine agreement given partial broadcast. 18(3):191–217, July 2005.

[CK88] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In pages 42–52, 1988.

[Cle86] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In pages 364–369, 1986.

[CM89] B. Chor and L. Moscovici. Solvability in asynchronous environments (extended abstract). In pages 422–427, 1989.

[Cré90] C. Crépeau. Verifiable disclosure of secrets and applications (abstract). In pages 150–154, 1990.

[Cré97] C. Crépeau. Efficient cryptographic protocols based on noisy channels. In pages 306–317, 1997.

[CvT95] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In pages 110–123, 1995.

[DPS$^+$12] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In pages 643–662, 2012.

[FGM$^+$01] M. Fitzi, J. A. Garay, U. M. Maurer, and R. Ostrovsky. Minimal complete primitives for secure multi-party computation. In pages 80–100, 2001.

[GIS$^+$10] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In pages 308–326, 2010.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In pages 218–229, 1987.

[IOS12] Y. Ishai, R. Ostrovsky, and H. Seyalioglu. Identifying cheaters without an honest majority. In pages 21–38, 2012.

[IOZ14] Y. Ishai, R. Ostrovsky, and V. Zikas. Secure multi-party computation with identifiable abort. In pages 369–386, 2014.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In pages 572–591, 2008.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In pages 20–31, 1988.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In pages 73–85, 1989.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In pages 84–93, 2005.

[SF16] G. Spini and S. Fehr. Cheater detection in SPDZ multiparty computation. In pages 151–176, 2016.

[Sha79] A. Shamir. How to share a secret. 22(11):612–613, November 1979.

[SSW10] A.-R. Sadeghi, T. Schneider, and M. Winandy. Token-based cloud computing. In A. Acquisti, S. W. Smith, and A.-R. Sadeghi, editors, *Trust and Trustworthy Computing*, pages 417–429, Berlin, Heidelberg. Springer Berlin Heidelberg, 2010.