Constructing Secure Multi-Party Computation with Identifiable Abort

Expanding Identifiable Abort against a Dishonest Majority

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

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

Abstract. Secure Multi-Party Computation (MPC) protocols based on *two-party* primitives like Oblivious Transfer have one severe drawback: the adversary can abort the protocol without repercussions if the majority of all parties are malicious. As a compromise between the drawback of Security with Abort and the unachievable notion of fairness, the notion of Identifiable Abort (IA) was formally introduced.

Given a broadcast channel, we tightly link the unanimous identifiability of any protocol to verifiable graph-theoretical properties of the Conflict Graph (CG). As an interesting side note, this technique suggests an inherent limitation on the overall number of parties as it connects identifying malicious parties in general in the IA-setting with solving the NP-hard problem of finding Minimum Vertex Covers.

We leverage the "Conflict Graph" in a concrete construction to give the first upper bound of the minimal setup size for n-party MPC with IA 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, i.e., for $t \coloneqq (n-k-3)$ corruptions we can construct n-party SFE from $(n-k-1)$-party SFE.

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

1 Introduction

Because of its power and wide range of application the Secure Multi-Party Computation (MPC) has been subject to extensive studies since the 1980's. If a majority of parties is honest, the situation is well understood as early results have established tight bounds whereas the situation becomes much harder when only a minority of parties are guaranteed to be honest. In particular, MPC suffers from a fundamental impossibility of Cleve [Cle86] which rules out *fairness* whenever a majority of parties are malicious. To circumvent this impossibility protocols that are secure against a dishonest majority, e.g. [IPS08], only provide *security with abort* where the adversary may abort the protocol, in particular after learning the output of the computation. However, this makes the protocols susceptible to Denial-of-Service attacks. To mitigate this undesired property the notion of Identifiable Abort (IA) has been considered by Aumann and Lindell [AL07] and Ishai, Ostrovsky, and Seyalioglu [IOS12] among others and formally introduced by Ishai, Ostrovsky, and Zikas [IOZ14]. Here the adversary may still abort the protocol but in the process the identity of (at least) one malicious party is revealed to all other parties. Intuitively, this enables the remaining parties to repeat the protocol without the perpetrator. As such, the notion of Identifiable Abort is the strongest one achievable for general MPC that is not limited to a small class of functionalities.

General MPC—in particular commitments—with composible security, e.g. in the Universal Composability (UC) framework [Can01], necessarily require a setup, typically a Common Reference String. While protocols fulfilling security with abort can be based on two-party setups like Oblivious Transfer (OT) [IPS08], the situation for Identifiable Abort is not thoroughly studied.

Related work. Fitzi, Garay, Maurer, and Ostrovsky [FGM$^+$01] initialized the study of the *minimal complete cardinality* for MPC, i.e. they give bounds for the minimal number of parties, that a setup needs to serve, vs. the fraction of corrupted parties. In particular, they showed that for an arbitrary number of corrupted parties the setup must include all parties. To this end they gave a *universal black-box* setup for n-parties that is oblivious to the actual evaluated function and the parties' inputs. Ishai, Ostrovsky, and Seyalioglu [IOS12] gave an n-party setup for MPC with Identifiable Abort based on *identifiable secret sharing* whose complexity is dependent only in the output of the evaluated function but not in the circuit size. They also show that pairwise functionalities, such as Oblivious Transfer, are insufficient even in presence of a broadcast channel if $t \geq 2n/3$. Ishai, Ostrovsky, and Zikas [IOZ14] formalized the notion of Identifiable Abort and gave a universal n-setup, *Correlated-Randomness*, which is oblivious to the evaluated function and the parties' inputs, akin to the universal black-box setup of [FGM$^+$01]. Baum, Orsini, and Scholl [BOS16] gave a practical protocol for MPC with IA. More recently, Baum, Orsini, Scholl, and Soria-Vazquez [BOS$^+$20] presented an efficient MPC protocol with IA in the dishonest majority setting which uses techniques similar to our formulation of the Conflict Graph. We see our work as a formal treatment of such conflict techniques to further their deployment in concrete MPC protocols. Even more recently, after the publication of the eprint version [BMM$^+$20] of this paper, Simkin, Siniscalchi, and Yakoubov [SSY21] showed how to construct the Correlated-Randomness setup from [IOZ14] for n-parties from a setup that provides Correlated Randomness to $n-1$ parties given that at most $t \leq n-2$ parties are corrupted. Being closest to our work in terms of feasibility results we want to elaborate on the differences and commonalities. Their approach uses a new variant of verifiable secret sharing based on [IOZ14] and uses broadcast as well. This improved our feasibility result slightly yet their results suffers from the same exponential blowup when applying the construction more than a constant number of times. We want to stress that our paper focuses more on the formalized application of graph theory to protocols with Identifiable Abort than on the actually obtained upper bound on the minimal complete cardinality itself. We feel that the graph-theoretical technique of Conflict Graphs (CGs)—while not conceptually new—provides a new angle on the problem of MPC with IA, and it can, in fact, be combined with verifiable secret sharing as demonstrated in our construction.

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 Section 2, we define the used notations and definitions, define our setting and introduce the functionalities we use, among others the new Fully Committed Oblivious Transfer variant. This new functionality is more thoroughly studied in Appendix A. We then follow with the main part of our work; we provide more detailed descriptions and detailed proofs of our Conflict Graph in Section 3. In Appendix B, we show that it can be easily constructed using only a broadcast channel. We give the concrete construction for expanding $n-1$-party SFE to n-party SFE in Section 4. In Section 5 we elaborate on SFE-expansion from smaller setups and efficiency issues. Finally, we conclude with a summary and an outlook in Section 6.

1.1 Contribution

Our three main contributions are:

Conflict Graph. We formalize a quite intuitive mechanism for cheater identification which we call the *Conflict Graph (CG)* and closely link the Identifiable Abort property of any MPC-protocol to verifiable graph-theoretical properties of the CG. The Conflict Graph is of theoretical interest on its own—we link Identifiable Abort to a problem related to the Vertex Cover problem which we call the *Intersecting Minimal Vertex Cover (IMVC)* problem and which is—in general—NP-hard. If the IMVC problem for CGs was shown to be efficiently solvable, then our restriction on the number of parties n could be lifted for our main construction (see Fig. 3). On the other hand, if the IMVC problem for CGs was proven to be (NP-)hard, this would imply an intrinsic upper bound of $n \in \mathcal{O}(\ln \lambda)$ parties for protocols with Identifiable Abort.

Technically, the Conflict Graph is a graph $G = ([n], E)$, where $[n]$ is the set of parties. G has a vertex for each party $\mathsf{P} \in [n]$. An edge $e = \{\mathsf{P}, \mathsf{P}'\} \in E$ stands for a publicly declared conflict

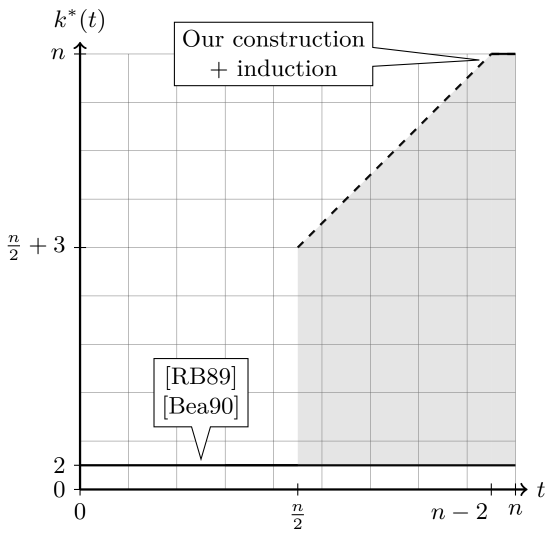


Fig. 1: Bounds of the minimal complete cardinality $k^*(n,t)$ with IA vs. *maximal* number of malicious parties t given broadcast. The grey area represents the possible region of $k^*(n,t)$. The dashed lines indicate our bounds ($\mathcal{O}(n)$ inductions for $n \in \mathcal{O}(\ln \lambda / \ln \ln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

between the two parties P and P'; we require that conflicts between honest parties never arise, hence an edge $e = \{\mathsf{P}, \mathsf{P}'\}$ is also a guarantee that P or P' (or both) are corrupted.

On an intuitive 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.

Reformulated Oblivious Transfer variant. The MPC-completeness of OT [Kil88; IPS08] is disproven in the setting of Identifiable Abort without broadcast [IOS12]. We thus reformulate Crépeau's *Committed* OT [Cré90; CvT95] in the multi-party setting, and prove it to be an MPC-complete building block in the setting of IA with broadcast. The Fully Committed Oblivious Transfer (FCOT) lets 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.

The main idea behind the use of the committed OT is the observation that in the IPS-compiler [IPS08] the abort occurs whenever an honest party notices some misbehaviour on some server. Though the honest party cannot (in the multi-party case) point with confidence to the actual cheater. However, if all OT where committed and witnessed by all other parties, then the honest party that noticed the cheating can simply challenge all parties to open all previous communication and thus all parties can retrace the computation and therefore identify the cheater or the falsely accusing party. See Appendix A.1 for the details.

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 instances of the $(n-1)$-party FCOT.

1.2 Setting

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 of choice ([Ajt96; Bon98; Reg05] to name only a few). We don't explicitly assume additional pairwise secure channels, since they can be emulated by hybrid functionalities of size ≥ 2.

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

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

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

The UC Framework. We perform our analysis in the Universal Composability (UC) framework [Can00; Can01], which is a strong version of simulation-based security [GM82; GM84; 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 participants inputs. The behavior of the trusted party is specified by a *functionality* $\mathcal{F}$. In the real world, all parties execute a protocol π, which is said to *realize* the functionality $\mathcal{F}$, if it can be shown to be indistinguishable from the ideal world. This requires a *simulator* who creates a transcript of an execution without knowing the parties inputs. More precisely, the transcripts of both worlds must be indistinguishable for any non-participant, even those who know the parties secret inputs. The transcript includes the output of all parties and the respective adversary. Indistinguishability 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 provides much stronger security guarantees than the standalone model, but comes with some restrictions; without a trusted setup, no protocol π can realize even SFE-incomplete functionalities such as Commitments [CF01], while computational constructions in the standalone model exist. 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 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[1] 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.

We adapt the view that the direct party-party communication can also be modeled as a two-party hybrid functionality. Therefore, in our hybrid protocol, honest parties have an authenticated connection with the hybrid functionalities; the adversary cannot manipulate messages from honest parties to hybrid functionalities and vice versa. This model corresponds to an adversary that might control the network but cannot fabricate false messages from honest parties.

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 against a dishonest majority [Cle86]. On the other extreme, the much weaker notion of Anonymous Abort (Security with Abort) leaves the adversary capable of stopping any computation without repercussions and thus is an undesirable property for many real-world scenarios.

[1] Version 20200212:021048

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 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[2]:

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

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

1.3 Overview

Conflict Graph. We introduce and analyze the Conflict Graph G and show how it can be used to link verifiable graph properties with the identification of disruptors in the field of Identifiable Abort. The Conflict Graph maintains the global (common) view of conflicts that arise between parties during a protocol execution.

Let π be an n-party protocol that uses the Conflict Graph. The CG of π is an undirected, simple graph $G = ([n], E)$ where $[n] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ is the set of parties of π. Intuitively, during the protocol execution a party P that notices any misbehavior of another party P' publicly announces that it is in conflict with P'. While honest parties may 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.

For protocols with Identifiable Abort, 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, in general, the precise condition of a conflict declaration based on protocol deviations is highly protocol-dependent; we defer the specifics of our constructions to Section 4. The formal description of the Conflict Graph functionality looks as follows:

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

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

- Upon first activation, initiate the set of conflict edges $E := \emptyset$.
- When receiving a message $(\texttt{conflict}, \mathsf{P}_i)$ from P_j, append the new conflict edge $\{\mathsf{P}_i, \mathsf{P}_j\}$ to the set of conflict edges E and send $(\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i)$ to the adversary.
- When receiving a message $(\texttt{query})$ from P_i, output G to P_i.

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

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

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

Keeping in mind that each conflict edge contains at least one corrupted party,[3] 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 P, which knows that itself is behaving honestly.

We define the $\mathcal{F}^n_{\mathsf{CG}}$ functionality merely for convenience and for structural clarity. It is easily constructed from a broadcast functionality (itself with IA) as it simply stores broadcasted information. See Appendix B for a formal construction.
In Section 5 we additionally provide an algorithm DeduceCG that—in some sense—completes the CG in a information-theoretic way. For clarity we refer to the CG returned by the functionality $\mathcal{F}^n_{\mathsf{CG}}$ as the *induced* CG G and to $G^* := \text{DeduceCG}(G, t)$ as the *deduced* CG. Since we only use the induced CG for efficiency considerations we defer further discussion to Section 5. 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 of which at most $0 \leq t < n$ are malicious. Let $G = ([n], E)$ be the Conflict Graph of a protocol π. 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 P is corrupted, then even non-participants can be convinced that party P is a disruptor.

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

Definition 3 (Biseparation). *A Conflict Graph $G = ([n], E)$ is called* biseparated*, iff there exists a subset $E' \subseteq E$ that forms a complete bipartite graph (biclique) on $[n]$.*

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 potentially much harder. See Section 5 for a discussion. We generally assume that protocols use the Conflict Graph in an abort-respecting way as specified in Definition 7. While this seems like a restriction at first, we show that it is not:

Lemma 1 (Informal version of Lemma 8). *Let n be the number of parties $[n]$ of which at most $0 \leq t < n$ are malicious. Let π be a protocol that securely UC-realizes a functionality $\mathcal{F}^n$ in the some M-hybrid model without $\mathcal{F}^n_{\mathsf{CG}}$. Denote by π' the $M \cup \{\mathcal{F}^n_{\mathsf{CG}}\}$-hybrid protocol that is identical to π with the addition that honest parties use $\mathcal{F}^n_{\mathsf{CG}}$ in an abort-respecting way.*

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

[3] This must be ensured by the protocol using the CG. In particular, our SFE-expansion fulfills this property.

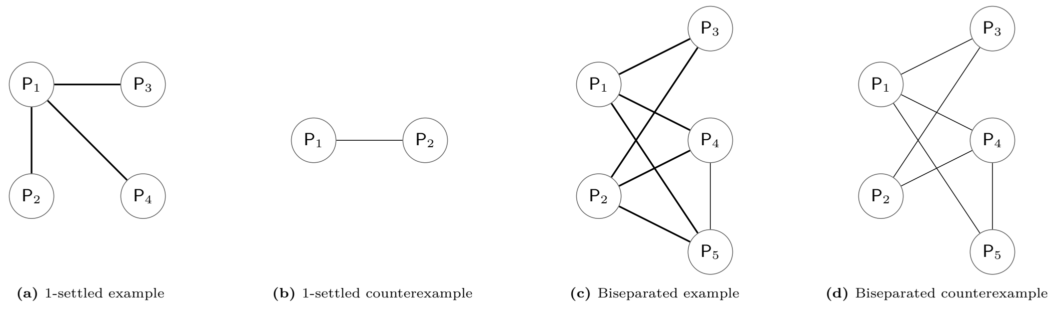


Fig. 2: Several (counter-)examples for the introduced conflict graph conditions. Thick lines are relevant for the respective property.

This lemma allows us to transfer results that only hold for protocols with abort-respecting CG usage to all protocols with Identifiable Abort. A consequence is that no adversary gains any advantage when honest parties use the CG in an abort-respecting way.

In Section 3 we elaborate on the relation between t-settledness and biseparation.

To give an intuition of the graph properties, a few example graphs are shown in Fig. 2. The graph in Fig. 2a has only one vertex cover of size 1, namely $\{\mathsf{P}_1\}$, thus P_1 is in all external explanations, making the graph 1-settled. Note that this graph is also biseparated. The graph from Fig. 2b on the other hand is not 1-settled; both $\{\mathsf{P}_1\}$ and $\{\mathsf{P}_2\}$ are valid Vertex Covers. However, biseparation trivially holds. Another example for biseparation is the Conflict Graph from Fig. 2c. 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 P_1 and P_2 are corrupted, thus allowing them to continue without them. However, if we remove the conflict between P_2 and P_5 (see Fig. 2d), the graph is no longer biseparated, thus P_5 would not agree to throw out P_2.

We now claim that biseparation of CG G and Identifiable Abort are closely related in the following sense.

- A biseparated CG G enables a protocol to abort consistently.
- Upon abort, the CG G of any protocol with IA must be biseparated.

The first direction is trivial: each party chooses its opposite partition as the disruptors. Since all honest parties are in one partition, the abort is consistent and the disruptors are indeed malicious. The other direction is not so easy to see.

Theorem 2 (Informal version of Theorem 6). *Let π be a protocol that securely UC-realizes a functionality $\mathcal{F}^n$ with Identifiable Abort in an $M \cup \{\mathcal{F}^n_{\mathsf{CG}}\}$-hybrid model. Upon abort, the Conflict Graph G of π must be biseparated.*

The proof of this theorem can be found in Theorem 6.

Remark 1. A Conflict Graph $G = ([n], E)$ 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.

For the other way, we require the property of biseparation for IA:

Remark 2. For any number of parties n, a biseparated CG implies IA.

Note that for $n \in \mathcal{O}(\log(\lambda))$, a t-separated CG can be efficiently converted to a *biseparated* CG, which relaxes the requirement of CG for IA even further. For larger n this is not evident, and indeed might actually be NP-hard. (See Section 5 for a discussion.)

SFE-Completeness of FCOT. We reformulate Crépeau'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 Lemmas 13 and 14). *For every number of parties n, the functionalities $\mathcal{F}^n_{\mathsf{SFE}}$ and $\mathcal{F}^n_{\mathsf{FCOT}}$ are equally powerful in the setting of IA.*

This implies that n parties can use SFE in order to obtain a FCOT and vice versa, that an FCOT hybrid suffices for SFE with IA.

SFE expansion We refer to all subfunctionalities that use the same set of parties as *instances* $\mathcal{F}$ of the same *instance class*, denoted by $[\mathcal{F}]$. That is, an instance class corresponds to a subset of parties $P \subseteq [n]$ and consists of all functionalities that interact with exactly P. Further we write that an instance class is considered aborted, if any instance of this class is aborted. This is because once the adversary aborted one instance by revealing a malicious party, the adversary can abort any other instance with the same party without revealing any new information. For functionalities of cardinality $n-1$ there are exactly n instance classes, one for each party that is excluded. For a given party $\mathsf{P}_i \in [n]$, we denote by $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ the instance class where all parties except for P_i participate.

Recall that the biseparation allows the protocol to abort consistently. We introduce three relevant lemmas, limiting the number of aborted instance classes of size $(n-1)$ in an n-party protocol before the graph becomes biseparated, effectively stating an upper bound on how many instance classes 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.

The most general version of our lemma regarding subfunctionality abort is the following:

Lemma 3 (Informal version of Lemma 9). *Let n be the number of parties $[n]$ of which at most $0 \leq t < n$ are malicious. Let π be an $\{\mathcal{F}^{n-1}, \mathcal{F}^n_{\mathsf{CG}}\}$-hybrid protocol that uses $\mathcal{F}^n_{\mathsf{CG}}$ in an abort-respecting way. If the adversary $\mathcal{A}$ aborts more than t instance classes of size $(n-1)$, then the Conflict Graph from $\mathcal{F}^n_{\mathsf{CG}}$ is biseparated.*

Note that, for simplicity only, we require protocols to use the CG in an abort-respecting way as described in Definition 7. A formal proof of the above lemma is deferred to Lemma 9. Here we only sketch the proof. Note that for any $i \in [n]$, after $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ has been aborted, the Conflict Graph G^* contains a biseparated subgraph over all parties except for P_i. We can investigate the complement graph $\overline{G}$. After an abort of $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$, the Complement Graph of the participants without P_i is split into two partitions, which are only connected via P_i in $\overline{G}$. We abuse notation and write $\overline{G} \setminus \mathsf{P}_i$ for the Complement Graph where P_i is removed. Say $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ was aborted by P_j. If the same party P_j aborts a second subfunctionality in which P_i is participating, its partition becomes completely disconnected in the Complement Graph, as P_i would also declare conflict with P_j. Thus, the adversary cannot "re-use" a party to abort a different instance class. Since there are only at most t corrupted parties, we have an upper bound on the number of instance classes 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 4 (Informal version of Lemma 10). *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 P_j which aborted $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ can only have two neighbors in the Complement Graph: party P_i, which did not witness the abort, and some other party P_k who aborted $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$. 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 P_l has an edge with the party that aborted $[\mathcal{F}]^{n-1}_{\mathsf{P}_l}$. 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 9 yields:

Lemma 5 (Informal version of Lemma 11). *Let $t < n$. Aborting more than $(n-2)$ instance classes $[\mathcal{F}]^{n-1}$ yields a biseparated Conflict Graph.*

Let P_i be an honest party. After an abort of $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$, 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 another instance class $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$ that omits party P_j is aborted, it can only be aborted by a subset of parties from the same partition as P_j, as otherwise, the graph becomes biseparated. This means that 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 reused. 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 4 and 5 play an essential role in providing a protocol for our main goal, namely SFE-expansion from $(n-1)$ parties to n parties. To achieve this goal, we use several lemmas, which we only sketch here. Our first lemma states that COM can be expanded from $(n-1)$ to n parties:

Lemma 6 (Informal version of Theorem 7). *For $t \leq (n-2)$, there exists a protocol π^n_{COM} in the $\{\mathcal{F}^{n-1}_{\mathsf{COM}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model that securely UC-realizes $\mathcal{F}^n_{\mathsf{COM}}$.*

Intuitively, we assume that there are $(n-1)$ instance classes $[\mathcal{F}]^{n-1}_{\mathsf{COM},\mathsf{R}_i}$, one for each omitted receiver R_i. The protocol π^n_{COM} lets the committer C input the same bit b into all subfunctionality instances $[\mathcal{F}]^{n-1}_{\mathsf{COM},\mathsf{R}_i}$. A receiver R_i only accepts the global unveil of a bit b, if the majority of commitments that R_i witnesses open to b, if at least three instance classes were not aborted, and if at most one of the unveils opens to to a different bit. Hence, a malicious committer C is forced to either input the same bit into all $\mathcal{F}^{n-1}_{\mathsf{COM},\mathsf{R}_i}$, 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 Theorem 7 that a corrupted C cannot successfully break the binding property.

The full protocol alongside the proof can be found in Theorem 7.

We summarize the above lemmas into a corollary, which limits the amount of subfunctionalities that can be aborted during the expansion of FCOT: Let S be the sender, R be the receiver and $(\mathsf{W}_1, \ldots, \mathsf{W}_{n-2})$ be the witnesses.

We classify the instances by the excluded parties. *Type-1* instance classes exclude an arbitrary witness, whereas *Type-2* instance classes exclude either the sender or the receiver. Then the following holds:

Corollary 1 (Informal version of Corollary 3). *If π realizes $\mathcal{F}^n_{\mathsf{FCOT}}$, then either at least two type-1 functionalities terminate successfully, or the Conflict Graph is biseparated.*

Lemma 10 with $t = (n-3)$ implies that either at least four instance classes of cardinality $(n-1)$ terminate successfully, or the Conflict Graph is biseparated. In the worst case, two of those could be of Type-2, leaving the remaining two to be of type-1.

Now, we use Corollary 3 to construct a protocol that securely realizes $\mathcal{F}^n_{\mathsf{FCOT}}$ from $\mathcal{F}^{n-1}_{\mathsf{FCOT}}$ and $\mathcal{F}^n_{\mathsf{COM}}$.

Lemma 7 (Informal version of Theorem 8). *For $t \leq (n-3)$, there is a protocol π^n_{FCOT} that UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^{n-1}_{\mathsf{FCOT}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model against any adversary that statically corrupts at most $t \leq (n-3)$ parties.*

The full proof of this lemma can be found in Theorem 8. The protocol we introduce there 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, two numbers $T, r \in \text{poly}(\lambda)$ define the total amount of shares to which the secrets are distributed. In each iteration, T is updated as number of remaining instance classes, which were not yet aborted. This is important, as aborted instance classes will be dropped and not used anymore.

A new round begins with the sender committing (globally) to a random mask $w \in \{0,1\}$ which it also uses for the additive secret sharing scheme to create T shares $(\mu^0_j)_{j=1..T}$ and $(\mu^1_j)_{j=1..T}$ of the *masked* messages $(m_0 \oplus w)$ and $(m_1 \oplus w)$, respectively. The mask hides the messages and allows the sender to decide when the receiver should learn the output. For $b \in \{0,1\}$, the sender uses the $(2r/3, r)$-threshold secret sharing scheme to create r shares $(\alpha^b_{j,i})_{i=1..r}$ of each additive share μ^b_j. Each of the obtained shares $\alpha^b_{j,i}$ is sent to $\mathcal{F}^n_{\mathsf{COM}}$; additionally, S distributes all the shares $\alpha^b_{j,i}$ using $(T \cdot r)$ $\mathcal{F}^{n-1}_{\mathsf{FCOT}}$ instances, with input $\big(\texttt{messages}, \alpha^0_{j,i}, \alpha^1_{j,i}\big)$. The receiver uses $\mathcal{F}^n_{\mathsf{COM}}$ to commit to the choice bit and sends the same choice bit c to all instances of $\mathcal{F}_{\mathsf{FCOT}}^{n-1}$. Witnesses only output their receipt, if all type-1 functionalities $\mathcal{F}_{\mathsf{FCOT}}^{n-1}$ have terminated successfully. If no subfunctionality instance was aborted, S unveils the mask w.

We use two methods to ensure that S provides correct shares: a cut-and-choose method lets each party broadcast a set of $r/10n$ indices for each subfunctionality. The sender then unveils the Commitments at those indices. Additionally when finding r shares which cannot be reconstructed to some additive share μ_j^b, R can accuse the sender of being a disrupter and demand opening of all sub-shares that lead to that share.

This protocol allows R to learn m_c, since it can obtain sufficiently many shares to reconstruct all T additive shares μ_j^c and the mask has been unveiled. Yet it does not allow R to learn m_{1-c}; the additive secret sharing scheme used for the creation of μ_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 $\mathcal{F}_{\mathsf{FCOT}}^{n}$, and $r/10$ additional shares, leaving him with $11r/10$ shares for each tuple (μ_j^0, μ_j^1). At least $2r/3$ shares are required for the construction of μ_j^c, yet reconstructing the whole tuple requires $4r/3$ shares.

At the same time, $\mathcal{F}_{\mathsf{COM}}^{n}$ ensures that both the inputs of S and 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 $\mathcal{F}_{\mathsf{FCOT}}^{n-1}$ instance classes.

With this, we arrive at our final theorem:

Theorem 3 (SFE expansion). *Let n be the number of parties of which at most $0 \le t \le (n-3)$ are malicious. The functionality $\mathcal{F}_{\mathsf{SFE}}^{n}$ can be UC-realized in the $\{\mathcal{F}_{\mathsf{SFE}}^{n-1}, \mathcal{F}_{\mathsf{BC}}^{n}\}$-hybrid model:*

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

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

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

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

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

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

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

By combining all lemmas, 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.

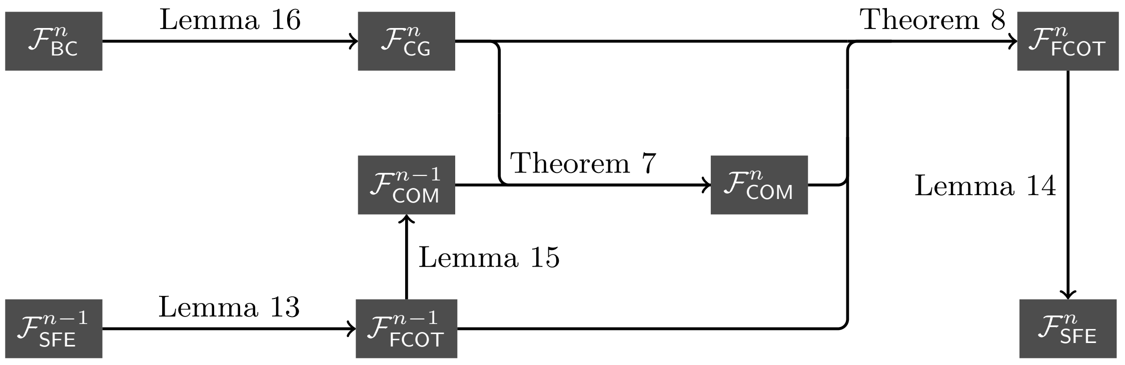
2 Definitions and Notation

We use the following conventions; also see the list of abbreviation and symbols.

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

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

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



$\mathcal{F}^n_{\mathsf{BC}}$
Lemma 16
$\mathcal{F}^n_{\mathsf{CG}}$
Theorem 8
$\mathcal{F}^n_{\mathsf{FCOT}}$
$\mathcal{F}^{n-1}_{\mathsf{COM}}$
Theorem 7
$\mathcal{F}^n_{\mathsf{COM}}$
Lemma 14
Lemma 15
$\mathcal{F}^{n-1}_{\mathsf{SFE}}$
Lemma 13
$\mathcal{F}^{n-1}_{\mathsf{FCOT}}$
$\mathcal{F}^n_{\mathsf{SFE}}$

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

Definition 4 ((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 5 (Minimal functionality). *We call a functionality $\mathcal{F}^m$ with cardinality m* minimal*, iff no functionality of lesser cardinality is complete.*

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 use secret sharing schemes in our constructions. Thus, we include a brief description:

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 $s = \bigoplus_{i=1}^m \sigma_i$.

2.1 Functionalities

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

Without writing it out explicitly each time, we generally assume that all of our functionalities are inherently *unfair*: public outputs are first sent to the corrupted parties. The adversary can then decide if honest parties should receive the output or not. However, withholding an output corresponds to an *abort* of the respective functionality; in our setting of Identifiable Abort, this 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 $\mathcal{F}^n_{\mathsf{SFE}}$

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

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

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

We denote by y_i the private output of a party P_i. Additionally, there is a common output $\varDelta$ which is obtained by all parties. We additionally use an input set $I \subseteq [n]$ 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.

$\mathcal{F}^n_{\mathsf{SFE}}$ is a versatile functionality, which can be used to even those functionalities,[4] where only one party has to provide input, and only an other party obtains output.

$\mathcal{F}^n_{\mathsf{SFE}}$ is not well-formed, meaning that it knows which of the parties are corrupted and which are honest. However, this is only required *upon abort*: Functionalities with IA need to know the set of malicious parties $C \subseteq [n]$ in order to verify that the set C' responsible for 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 For our constructions, we use a variant of the One-to-Many Commitment from [CLO+02], with slight modifications to make them suitable for Identifiable Abort. In this paper, we call the One-to-Many Commitment a Global Commitment.

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

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

- When receiving $(\texttt{commit}, m)$ with $m \in \{0,1\}$ from party 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 C.
- When receiving $(\texttt{unveil})$ from party 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 $\mathcal{F}^n_{\mathsf{COM}}$ outputs $\big(\texttt{abort}, C'\big)$ to all parties, and then terminates.

Beyond the guarantees that the Global Commitment inherited from two-party Commitments, namely the *binding* and *hiding* properties, $\mathcal{F}^n_{\mathsf{COM}}$ additionally ensures *consistency* in that the committer C is committed to *the same* bit against all receivers.

We also make use of the ideal broadcast functionality from [CLO+02], with modifications to support Identifiable Abort.

[4] Such as *commitments*

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

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

- When receiving a message $(\texttt{input}, m)$ with $m \in \{0,1\}^*$ from party 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 $\mathcal{F}^n_{\mathsf{BC}}$ outputs $\big(\texttt{abort}, C'\big)$ to all parties, and then terminates.

We consider $\mathcal{F}^n_{\mathsf{BC}}$ to be a relatively weak functionality; we use it primarily to realize the Conflict Graph functionality $\mathcal{F}^n_{\mathsf{CG}}$ from Section 3. We provide an instantiation of $\mathcal{F}^n_{\mathsf{CG}}$ using $\mathcal{F}^n_{\mathsf{BC}}$ in Appendix B, thus proving that $\mathcal{F}^n_{\mathsf{CG}}$ too is a relatively weak setup.

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

Furthermore, note that it is possible to construct a broadcast $\mathcal{F}^n_{\mathsf{BC}}$ from a Global Commitment $\mathcal{F}^n_{\mathsf{COM}}$, by letting the sender S *commit* and immediately *unveil* the message.

Fully Committed Oblivious Transfer Next, we introduce our OT variant called Fully Committed Oblivious Transfer (FCOT), which extends *normal* OTs in two ways: 1. it includes $n-2$ witnesses, which obtain a *receipt* if the message has been transferred successfully, 2. the sender S is committed to both messages m_0 and m_1, and 3. the receiver R is committed to c.

We rigorously use for our construction in Section 4:

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

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

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

When receiving $\big(\texttt{abort}, C'\big)$ from $\mathcal{A}$ with $C' \subseteq C$, then $\mathcal{F}^n_{\mathsf{FCOT}}$ outputs $\big(\texttt{abort}, C'\big)$ 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.

Our ideal functionality $\mathcal{F}^n_{\mathsf{FCOT}}$ is an extension of conventional OT, which turns out to be quite 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 the construction of [IPS08], a party P notices when another party P'

misbehaves towards P. However, a third party P^* need not necessarily notice such misbehavior of P' towards P. Hence parties cannot convince others of the identity of a disruptor, leaving those constructions only with Anonymous Abort. In Appendix A.1, 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 thanks to their secret sharing.

3 Conflict Graph

We elaborate on our CG technique introduced 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.

While not conceptionally new, we provide a formal framework to facilitate analysis and design of MPC protocols with Identifiable Abort. We introduce an ideal functionality $\mathcal{F}^n_{\mathsf{CG}}$ which administrates accusations of misbehavior during the execution. Since the functionality $\mathcal{F}^n_{\mathsf{CG}}$ merely abstracts away the publication and retaliation of conflicts via broadcasts, we postpone the detailed construction to Appendix B and focus on its usage here. Before going into detail about the abort-respecting usage of $\mathcal{F}^n_{\mathsf{CG}}$ we want to reiterate the relevant graph properties for IA.

The novel utility of the CG lies in the connection of (unanimous) identifiability of malicious parties 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 us briefly recall the two properties:

- t-settledness: Intuitively there is at least one fixed party present in all possible explanations (MVCs) of size $\leq t$. This means that t-settledness of a Conflict Graph is a minimal requirement to identify at least one malicious party with absolute certainty. Hence, this suffices to convince both participants and outsiders that a party is a disruptor.
- Biseparation: The graph contains two partitions, where each party in one partition is in conflict with every party from the other partition. Given a biseparated graph, participants can identify disruptors as all parties in the opposite partition, but outsiders might not know which partition contains honest parties.

We now introduce three important results consolidating the relevance of the Conflict Graph in the IA-setting, informally stating the following: for any protocol π that implements a functionality with IA, (i) a biseparated Conflict Graph G of π implies that the honest parties of π can identify disruptors, thus enabling IA, (ii) Identifiable Abort of π results in a biseparated CG, and (iii) π can be augmented with the CG-functionality $\mathcal{F}^n_{\mathsf{CG}}$. By combining Items (i) and (ii) we can conclude that, in a sense, Identifiable Abort and biseparation of G are equivalent.

Item (iii) enables us to transform any secure protocol into one which uses the CG such that our results apply. This allows us to only quantify over protocols that use the CG in an abort-respecting way.

Definition 7 (Abort-Respecting usage of CG). *A protocol π in the $\mathcal{F}^n_{\mathsf{CG}}$-hybrid model uses the Conflict Graph in an* abort-respecting *way, if the following conditions are fulfilled:*
(1) It is guaranteed that no honest–honest–conflicts occur.
(2) Whenever a subfunctionality with parties $S \subseteq [n]$ is identifiably aborted with disruptors $D \subset S$, the honest parties in S declare conflicts with D.
(3) Whenever (honest) parties abort the protocol by identifying disruptors $D \subset S$ they first call $\mathcal{F}^n_{\mathsf{CG}}$ with each $(\texttt{conflict}, \mathsf{D})$ *s.t.* $\mathsf{D} \in D$ *to declare a conflict with D.*

While Item (1) is automatically fulfilled when only considering subfunctionality abort from Item (2) due to the correctness of the respective subfunctionality with respect to aborts, this property is not so obvious for protocol-specific conflict declaration resulting from a deviation in the outer (n-party) protocol—which can occur if a party sends a message that notably differs from any message an honest party would send.

This scenario is caught by Item (3). To ensure that a conflict is only issued if one party really is corrupt, we demand honest parties to only declare a conflict if they have absolute certainty that the other party is corrupt. This is modeled by letting them *only* issue conflicts if they also abort.

For Item (2) note that this causes the subgraph on the participating parties S to be *biseparated*; similar to the uni-/multi-abort, honest parties in S declare conflict with D and all parties that do not declare conflict with D.

Unfortunately, our results do not apply to arbitrary protocols that are not in the $\mathcal{F}^n_{\mathsf{CG}}$-hybrid model or that use $\mathcal{F}^n_{\mathsf{CG}}$ arbitrarily, but only for those that are abort-respecting according to Definition 7. However, we show that this is *not* a restriction: each protocol can be augmented with $\mathcal{F}^n_{\mathsf{CG}}$ in a trivial manner such that our results apply to the augmented protocol.

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

Proof. The intuition of this proof is that the information provided by $\mathcal{F}^n_{\mathsf{CG}}$ is only useful for honest parties. In other words, to prove our claim we show that the environment can infer the Conflict Graph from its own behavior, thus proving that $\mathcal{F}^n_{\mathsf{CG}}$ 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 π' from simulated ones. We show that the interaction of $\mathcal{Z}'$ with $\mathcal{F}^n_{\mathsf{CG}}$ can be simulated correctly to ensure that no efficient distinguishing is possible.

First, note that $\mathcal{Z}'$ can obtain information from $\mathcal{F}^n_{\mathsf{CG}}$ by letting the dummy adversary issue a query to $\mathcal{F}^n_{\mathsf{CG}}$ in the name of any corrupted party. As the definition of $\mathcal{F}^n_{\mathsf{CG}}$ treats queries independently of the parties corruption state this yields the correct Conflict Graph. Hence $\mathcal{Z}'$ cannot directly be used to distinguish π from its simulations.

To provide $\mathcal{Z}'$ with a consistent view of the Conflict Graph we define a new environment $\mathcal{Z}$ which internally runs $\mathcal{Z}'$. The new environment simulates $\mathcal{F}^n_{\mathsf{CG}}$ and supplies $\mathcal{Z}'$ with the necessary information. To this end, $\mathcal{Z}$ starts with an empty CG $G \coloneqq ([n], E)$ with $E \coloneqq \emptyset$. For subfunctionality abort according to Item (2), whenever $\mathcal{Z}$ obtains $(\mathtt{abort}, D)$ from any corrupted party for any subfunctionality on $S \subseteq [n]$, the environment $\mathcal{Z}$ biseparates its simulated CG G on S between D and $S \setminus D$. Note that this covers both subfunctionality aborts according to Items (2) and (3), the latter implying $S = [n]$. When $\mathcal{Z}'$ queries $\mathcal{F}^n_{\mathsf{CG}}$, the outer environment $\mathcal{Z}$ supplies $\mathcal{Z}'$ with its CG G. The simulated CG is identical to the one provided by $\mathcal{F}^n_{\mathsf{CG}}$ in π' by requirement on π'. If a corrupted party declares a conflict, then $\mathcal{Z}$ also incorporates this conflict into its simulated CG.

Using the inner environment $\mathcal{Z}$, the outer environment $\mathcal{Z}'$ can distinguish π from its simulations; this is a contradiction to our requirement that $\pi \geq \mathcal{F}^n$. □

This preliminary result already provides a quite interesting high-level interpretation: honest parties have no disadvantage in using the Conflict Graph according to Definition 7; at worst the malicious parties obtain information they already know. 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 helps honest parties.

Now that we have seen that *all* protocols can easily be augmented with $\mathcal{F}^n_{\mathsf{CG}}$, we shall proceed to show that Identifiable Abort corresponds to biseparation of the corresponding Conflict Graph.

Theorem 6 (Biseparated Identifiable Abort). *Let n be the number of parties of which at most $0 \leq t < n$ are malicious. Let π be a protocol that securely UC-realizes a functionality $\mathcal{F}^n$ with Identifiable Abort in an $\{\mathcal{F}^m, \mathcal{F}^n_{\mathsf{CG}}\}$-hybrid model with $m \leq n$. Let π use $\mathcal{F}^n_{\mathsf{CG}}$ in an abort-respecting way according to Definition 7. Upon abort, the Conflict Graph G of π 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. For the sake of contradiction, let π be a protocol that was aborted, where honest parties settled on some party P, but where the Conflict Graph is *not* biseparated. Let $\mathcal{Z}$ be the environment that caused the CG. We prove that the lack of biseparation allows for a different environment $\mathcal{Z}'$ that causes the same CG: it lets honest parties output P, but where P is *honest.* Since no simulator can abort $\mathcal{F}^n$ with an honest party, this directly contradicts the presupposition that the protocol π securely realizes $\mathcal{F}^n$.

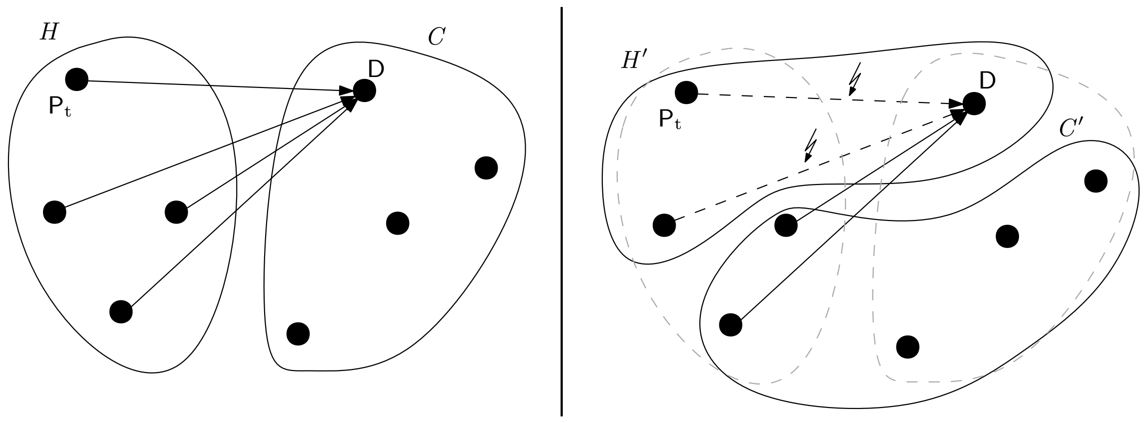


Fig. 4: Visualization of an exemplary set of corrupted parties and an assumed honest parties agreement on D for an environment $\mathcal{Z}$ on the left. The right side shows another, specifically constructed set of corrupted parties by an environment $\mathcal{Z}'$ which leads to honest-honest-accusations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

All honest parties unanimously settling for a disruptor P against environment $\mathcal{Z}$ implies that there must be some mechanism they used. We model this mechanism as a *selector function* S_π; honest parties use the output of S_π to agree on a common disruptor D.

We show that such an agreement, hence such a function, cannot exist: the original environment $\mathcal{Z}$ caused the given CG-transcript τ where party P was accused of cheating. We construct an environment $\mathcal{Z}'$ which creates *the same* Conflict Graph transcript, but in a setting where P is honest.

Let $\mathrm{S}_\pi \colon [n] \times \tau \to 2^{[n]}$ with $(\mathsf{P}, \tau) \mapsto D$ for $D \subseteq [n]$ be the selector function that specifies the identified disruptor (set) for each (honest) party P, given the set of inputs to $\mathcal{F}^n_{\mathsf{CG}}$ as τ. Without loss of generality, let $|D| = 1$. The selector S_π 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 denote by C the set of parties corrupted by $\mathcal{Z}$. Correctness dictates that $\mathsf{P} \in C$, meaning that the disruptor all honest parties agreed on really is malicious; otherwise, we are done. We construct a different environment $\mathcal{Z}'$ that causes the same transcript τ of the Conflict Graph G. By definition of S_π, this would lead to an identification of the same party P.

$\mathcal{Z}$ corrupted up to t parties $C \in M(G, t)$. Denote the complement set of honest parties by $H := [n] \setminus C$.

Without losing generality of our proof,[5] we can assume that the environment $\mathcal{Z}$ only lets corrupted parties $\mathsf{D} \in C$ broadcast conflicts as a retaliation; that is, it broadcasts conflicts $(\texttt{conflict}, \mathsf{P})$ against an honest party P in the name of D only after P has publicly declared a conflict $(\texttt{conflict}, \mathsf{D})$.

Since we assume G to be not biseparated, there exists another possible corrupted set of parties (a different MVC on G) $C' = [n] \setminus H' \in M(G, t)$ with $C \cup C' \neq [n]$, or equivalently $H \cap H' \neq \emptyset$. If $\mathsf{D} \in \mathrm{S}_\pi(\mathsf{P}, \tau) \subseteq C \cap H'$ for all $\mathsf{P} \in H$, then π cannot securely realize $\mathcal{F}^n$. Here the corrupted MVC C' is chosen such that the identified party against $\mathcal{Z}$ is explicitly excluded.

Thus, we can fix a *target* party P_t in $H \cap H'$, which by requirement will select $\mathsf{D} \in \mathrm{S}_\pi(\mathsf{P}_\mathrm{t}, \tau)$ against both environments, despite D being honest when playing with $\mathcal{Z}'$. The transcripts τ against both environments are equal, hence the selection function S_π has the same output distribution against both environments; we can safely assume that the two selected parties must also be equal. Fig. 4 depicts an exemplary visualization of the corruption sets.

Against the environment $\mathcal{Z}'$, an honest party is identified by at least one honest party. 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 $\mathsf{D} \in C^0 \cap H^1$, π cannot securely realize $\mathcal{F}^n$.

Thus, we have to show that such an MVC actually exists. If it does not exist, then for all feasible explanations $C' \in M(G, t)$, it holds that $C \cup C' = [n]$, or if for all $\mathsf{P}_\mathrm{t} \in H$, it holds that

[5] Our proof works by contradiction, i.e., we argue that there *exists* such an environment $\mathcal{Z}$.

$\mathrm{S}_\pi(\mathsf{P}_t, \tau) \subseteq H \cup C'$. The former is equivalent to $C' \supseteq H$, which contradicts the initial assumption that G is not biseparated: If $C' = H = [n] \setminus C$ are both MVCs, this would imply a biseparated Conflict Graph with partitions C and C'. If $C' \supsetneq H$, it would follow that H is a valid explanation, that is, $H \in M(G,t)$, but of smaller size. Hence, we get that C' 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 a set of corrupted parties $C' \in M(G,t)$ that fulfills $C \cup C' \neq [n]$ and $\mathrm{S}_\pi(\mathsf{P}, \tau) \subseteq C \cap H'$ for all $\mathsf{P} \in H$, we can define an environment $\mathcal{Z}'$ that corrupts C' and acts such that it produces τ before the abort.

We conclude our proof by showing that $\mathcal{Z}'$ really can create an equivalent transcript τ. The transcript τ of the Conflict Graph $G = ([n], E)$ is essentially an ordered list of (directed) edges $(e_j)_{j=1\ldots m}$. 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 $\mathcal{Z}$, the environment $\mathcal{Z}'$ behaves as follows:

If $u_j \in C \cap C'$**,** $\mathcal{Z}$ and $\mathcal{Z}'$ behave identically.

If $u_j \in C \cap H'$**,** then by assumption, $\mathcal{Z}$ 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 $\mathcal{Z}'$ lets v_j broadcast $(\texttt{conflict}, u_j)$ such that the retaliation e_j follows subsequently from the honest party u_j.

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

If $u_j \in H \cap H'$**,** $\mathcal{Z}$ and $\mathcal{Z}'$ behave identically.

The transcript produced by $\mathcal{Z}'$ is therefore identical to the one from $\mathcal{Z}$. □

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

Before we explain our proof for SFE-expansion, we need three additional lemmas. Each provides a limit on the maximum number of hybrid instance classes that can be aborted; this implies a guarantee that either some instance classes have to terminate successfully, or a malicious party is exposed. Using this guarantee, we provide a protocol that uses n-party broadcast to expand $(n-1)$-party Global Commitments $\mathcal{F}^{n-1}_{\mathsf{COM}}$ to n-party Global Commitments $\mathcal{F}^{n}_{\mathsf{COM}}$. We then use this n-party commitment as a tool in the expansion of FCOT from $(n-1)$ parties to n parties.

Lemma 9 (General subfunctionality abort). *Let n be the number of parties of which at most $0 \leq t < n$ are malicious. Let π be an $\left\{[\mathcal{F}]^{n-1}, \mathcal{F}^{n}_{\mathsf{CG}}\right\}$-hybrid protocol that uses the Conflict Graph in an abort-respecting way according to Definition 7. If the adversary $\mathcal{A}$ aborts more than t subfunctionality instance classes of cardinality $(n-1)$, then the Conflict Graph from $\mathcal{F}^{n}_{\mathsf{CG}}$ is biseparated.*

Proof. Denote the set of n parties by $[n] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. 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 instance classes, there is a set of parties 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 instance classes are aborted.[6] Since honest parties are never in conflict, their mutual edges are never removed. We only consider instance classes of cardinality $(n-1)$; thus, we have one subfunctionality instance class for each excluded party $\mathsf{P} \in [n]$.

We call the set of corrupted parties that aborts a subfunctionality instance class the *disruptor set D*. By D_i we denote the disruptor set for the subfunctionality instance class that excludes 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

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

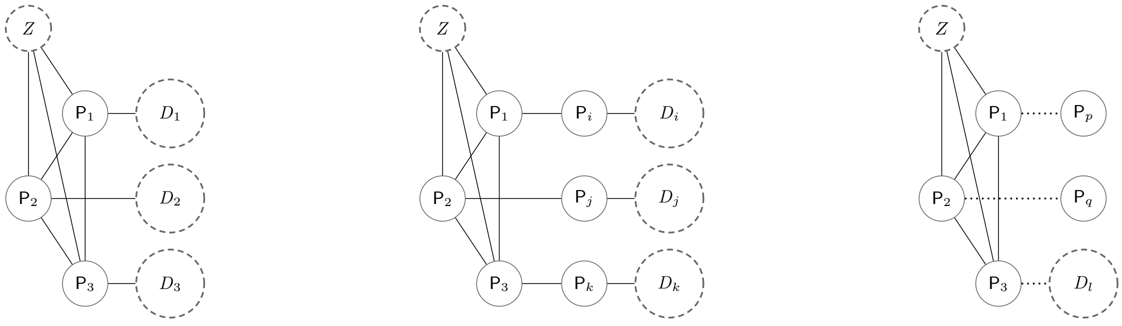


Fig. 5: Complementary Conflict Graph for the abort strategy with three honest parties Left: Result of aborting $[\mathcal{F}]^{n-1}_{\mathsf{P}_1}$, $[\mathcal{F}]^{n-1}_{\mathsf{P}_2}$ and $[\mathcal{F}]^{n-1}_{\mathsf{P}_3}$. Middle: Same graph after additionally aborting $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$, $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$ and $[\mathcal{F}]^{n-1}_{\mathsf{P}_l}$. Right: The P_3 branch can never be terminated.

two different disruptor sets D_i and D_j, then this intersection $I_{ij} := D_i \cap D_j$ would be in conflict with all parties who participated in $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ and with the ones who participated in $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$, 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 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 1). Therefore, any instance class $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ 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 $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ for some $\mathsf{P}_i \in D_1$. This causes all remaining parties $\mathsf{P}_j \in D_1 \setminus (\{\mathsf{P}_i\} \cup D')$ to declare a conflict with D'. Thus, those parties would be separated from D'. Since P_i was excluded from this instance class, 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 $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$ 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 instance class corresponding to the leaf node of each branch; there would have to be some party left that could abort the instance class, which would then cause a conflict, thus separating a subset of the branch from the rest. As a consequence, at least $(n-t')$ instance classes must succeed. The adversary can thus abort at most t' instance classes, before the Conflict Graph becomes biseparated. □

We consider two special cases of Lemma 9: one with 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 instance classes of cardinality $(n-1)$ can be aborted when using the Conflict Graph technique. For this case, Lemma 9 tightens to:

Lemma 10 (Strong subfunctionality abort). *Let $0 \leq t \leq (n-3)$. If t or more instance classes of cardinality $(n-1)$ are aborted, then the Conflict Graph is biseparated.*

Proof. It suffices to consider the case for t aborted instance classes. Each of the t disruptors D' can have at most two edges in $\overline{G}$, namely the party P_i that was omitted in the functionality $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ that D' aborted and the party P_j that disrupted $[\mathcal{F}]^{n-1}_{\mathsf{D}'}$. However, honest parties P_k can still have up to three neighbors in $\overline{G}$, namely two actual honest parties and one disruptor who aborted $[\mathcal{F}]^{n-1}_{\mathsf{P}_k}$. 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 9 yields:

Lemma 11 (Weak subfunctionality abort). *Let $0 \leq t < n$. Either at most $(n-2)$ instance classes of cardinality $(n-1)$ are aborted, or the Conflict Graph is biseparated.*

Proof. We only proof the case of a single honest party P. If more parties are honest, less instance classes can be aborted.

Denote the set of parties by $[n] = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$; w.l.o.g., assume that $\mathsf{P} = \mathsf{P}_1$, meaning that P_1 is honest. Furthermore, denote the corrupted parties as C.

Let $[\mathcal{F}]^n_{\mathsf{P}_1}$ be a $(n-1)$-party functionality that is executed as part of a bigger n-party computation.[7] Upon abort of $[\mathcal{F}]^n_{\mathsf{P}_1}$, the subgraph on $[n] \setminus \{\mathsf{P}_1\}$ must be biseparated. The only way this does not result in a biseparated Conflict Graph is if is all other parties are malicious. If one partition is of size $> t$, the excluded party P_1 can efficiently identify the settled set, causing a completely biseparated Conflict Graph. If the subgraph on $[n] \setminus \{\mathsf{P}_1\}$ only biseparated, we call the two partitions S_0 and S_1. We now focus our attention to the complement Conflict Graph, $\overline{G} = ([n], \overline{E})$. 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, P_1 is still connected to both partitions. The instance class of 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 P_i, whose hybrid functionality $[\mathcal{F}]^{n-1}_{\mathsf{P}_i}$ cannot be aborted, without creating a biseparated Conflict Graph. □

We now have proven 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 $\mathcal{F}^{n-1}_{\mathsf{COM}}$ to $\mathcal{F}^{n}_{\mathsf{COM}}$.

Our proof for SFE-expansion uses those three lemmas. Recall that it follows from Lemmas 9 to 11, that the Conflict Graph is biseparated when the adversary aborts more than $\min(t, n-2)$ subfunctionalities.

With those limits, we are able to expand the (SFE-incomplete) functionality Global Commitment (GCOM) from $\mathcal{F}^{n-1}_{\mathsf{COM}}$ to $\mathcal{F}^{n}_{\mathsf{COM}}$.

Theorem 7 (COM expansion). *Let n be the number of parties of which at most $0 \leq t \leq (n-3)$ are malicious. There exists a protocol π in the $\{\mathcal{F}^{n-1}_{\mathsf{COM}}, \mathcal{F}^{n}_{\mathsf{BC}}\}$-hybrid model that securely UC-realizes $\mathcal{F}^{n}_{\mathsf{COM}}$:*

$$\{\mathcal{F}^{n-1}_{\mathsf{COM}}, \mathcal{F}^{n}_{\mathsf{BC}}\} \rightsquigarrow \mathcal{F}^{n}_{\mathsf{COM}} \tag{2}$$

Proof. We only consider bit-commitments, which can be canonically extended to string-commitments. Let the set of parties be $[n] = \{\mathsf{C}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$ and let b be the bit that the committer C commits to. We present a protocol that uses $(n-1)$ COM instance classes $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$ for each excluded receiver R_l. Additionally, it uses the Conflict Graph $\mathcal{F}^{n}_{\mathsf{CG}}$ which can be realized using $\mathcal{F}^{n}_{\mathsf{BC}}$. Denote C's input to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$ by b_l and the abort set for $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$ by D_l. Let I_0 be the set of instance classes where $b_l = 0$, let I_1 be the set of instance classes where $b_l = 1$, and let $I_\times$ be the set of subfunctionalities that have previously been aborted. Note that $\mathcal{F}^{n}_{\mathsf{CG}}$ ensures that $I_\times$ is public knowledge; the abort of a subfunctionality $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$ results in a biseparated subgraph with vertices $[n] \setminus \{\mathsf{R}_l\}$. If a subgraph is biseparated, the corresponding instance class is considered aborted. It holds that $|I_0| + |I_1| + |I_\times| = (n-1)$, since there are $(n-1)$ instance classes $[\mathcal{F}]^{n-1}_{\mathsf{R}_1}$ through $[\mathcal{F}]^{n-1}_{\mathsf{R}_n}$.

In the following, we give a description of the protocol. Let b be the bit C wants to commit to. For runtime restrictions, we assume that the unary security parameter 1^λ is also input.

[7] This implies that despite all $(n-1)$ participants being malicious and thus the functionality technically never being called, P_1 knows that this functionality is executed, and aborts have to be reported correctly, or the protocol is considered to be terminated successfully.

Protocol:

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

   If $|I_\times| \geq t$**,** then the Conflict Graph is biseparated due to Lemma 10. R_l aborts with the identified set of parties.

   Otherwise, R_l outputs $(\texttt{receipt commit})$.
3. On input $(\texttt{unveil})$ from $\mathcal{Z}$ to C, C sends $(\texttt{unveil})$ to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$ for all $l \in \{1, \ldots, n-1\}$.
4. Once R_l has received output from all $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$ for $j \neq l$, R_l sends $(\texttt{receipt}, \mathsf{R}_l)$ to $\mathcal{F}^n_{\mathsf{BC}}$.
5. Once R_l has received $(\texttt{receipt}, \mathsf{R}_j)$ from all R_j for $j \neq l$ via $\mathcal{F}^n_{\mathsf{BC}}$, R_l checks $|I_\times|$.

   If $|I_\times| \geq n-3$**,** then the Conflict Graph is biseparated due to Lemma 10. R_l aborts with the identified set of parties.

   If $|I_\times| \leq n-4$**,** R_l checks the unveil messages:

   Since $n - |I_\times| \geq 4$ each receiver gets at least three messages. Recall that there is also $[\mathcal{F}]^{n-1}_{\mathsf{C}}$ which is not used. If all but one unveilings are consistent with b', R_l outputs $(\texttt{unveil}, b')$.

   Otherwise, R_l sends $\{\texttt{conflict}, \mathsf{C}\}$ to $\mathcal{F}^n_{\mathsf{CG}}$ and outputs $(\texttt{abort}, \mathsf{C})$.

In our synchronous model, messages sent by a Sender are guaranteed to be received by the dedicated Receiver. If 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: (1) the Conflict Graph is already biseparated after Step 2, which implies that all honest parties have aborted; or (2) the missing message's sender R_j must be corrupted. Hence a missing message from R_j is considered an abort of $\mathcal{F}^n_{\mathsf{BC}}$ from R_j. Since this biseparates the entire Conflict Graph, the $\mathcal{F}^n_{\mathsf{COM}}$ functionality is aborted.

The intuition behind the protocol is the following: The committer 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 Lemma 10, 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 three are left, the Conflict Graph has sufficiently many edges that a identification is possible.

We note that our strategy does not work directly for $t \geq (n-2)$ because Lemma 11 only guarantees that two instance classes must work. Therefore one receiver may only receive one unveiled bit.

Proving security of this protocol is straightforward. First note that in every case the behavior of honest receivers in unambiguous, meaning consistent among receivers. 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 $(\texttt{receipt commit})$ from $\mathcal{F}^n_{\mathsf{COM}}$, the simulator gives local input to all uncorrupted simulated parties as follows: The simulator gives the uncorrupted C local input $(\texttt{commit}, 0)$, hence C inputs $(\texttt{commit}, 0)$ into all $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$.

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

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

On input $\big(\texttt{abort}, C'\big)$ from $\mathcal{Z}$ for $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$, the simulator sends $\big(\texttt{abort}, C'\big)$ to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$.

If the (simulated) Conflict Graph becomes biseparated, the simulator aborts $\mathcal{F}^n_{\mathsf{COM}}$ 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 C to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$, the simulator lets C pass $(\texttt{commit}, m_l)$ on to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$. After receiving local input for all (non-aborted) subfunctionalities, the simulator lets C input $(\texttt{commit}, m)$ into $\mathcal{F}^n_{\mathsf{COM}}$ where m is the majority of all m_l. If there is no majority then C inputs a random bit m. The inputs $(\texttt{commit}, m_l)$ are naturally passed to the simulated $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$.

On input $(\texttt{unveil})$ from corrupted C to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$, the simulator lets C forward $(\texttt{unveil})$ to $[\mathcal{F}]^{n-1}_{\mathsf{R}_l}$.

On input $(\texttt{receipt commit})$ from corrupted R_l to $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$, the simulator forwards $(\texttt{receipt commit})$ in the name of 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. □

The COM-expansion lemma plays into the next lemma by ensuring consistency across multiple FCOT instance classes. Now we have all the tools required to investigate the expansion of FCOT. We denote by S the sender, by R the receiver and by $(\mathsf{W}_1, \ldots, \mathsf{W}_{n-2})$ the witnesses. We categorize the used instance classes in two categories: *Type-1* instance classes consist of functionalities where both S and R participate, that is, all $[\mathcal{F}]^{n-1}_{\mathsf{W}_i}$ for any $i \in \{1, \ldots, n-2\}$. *Type-2* instance classes are $[\mathcal{F}]^{n-1}_{\mathsf{S}}$ and $[\mathcal{F}]^{n-1}_{\mathsf{R}}$.

Corollary 3. *Let π be a protocol that securely realizes $\mathcal{F}^n_{\mathsf{FCOT}}$. At least two type-1 functionalities cannot be aborted, otherwise the Conflict Graph is biseparated.*

This follows from Lemma 10 with $t = (n-3)$, which states that only strictly less than $(n-3)$ instance classes of cardinality $(n-1)$ can be aborted without producing a biseparated Conflict Graph. Conversely, at least four instance classes of cardinality $(n-1)$ must succeed, two of which may be $[\mathcal{F}]^{n-1}_{\mathsf{S}}$ and $[\mathcal{F}]^{n-1}_{\mathsf{R}}$. Hence the remaining two must be of type-1.

Using Corollary 3, which states that with three or more honest parties, there must be two subfunctionalities that terminate or else the Conflict Graph becomes biseparated, we can now prove the final step missing in Fig. 3; namely, the actual expansion of $\mathcal{F}^n_{\mathsf{FCOT}}$.

Theorem 8 (FCOT expansion). *Let n be the number of parties of which at most $0 \le t \le (n-3)$ are malicious. There is a protocol π^n_{FCOT} that UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$ in the $\{\mathcal{F}^{n-1}_{\mathsf{FCOT}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model:*

$$\{\mathcal{F}^{n-1}_{\mathsf{FCOT}}, \mathcal{F}^n_{\mathsf{BC}}\} \rightsquigarrow \mathcal{F}^n_{\mathsf{FCOT}} \tag{3}$$

Proof. We describe the protocol π^n_{FCOT}. Denote the parties as sender S, receiver R and witnesses W_1 through W_{n-2}.
The protocol is iteration-based. Let $I_\times$ be the set of type-1 subfunctionalities that have been aborted. There are $T := (n-2-|I_\times|)$ type-1 subfunctionalities that have **not** yet been aborted. In each iteration of the protocol enumerate these as $\left([\mathcal{F}]^{n-1}_{\mathsf{W}_1}, \ldots, [\mathcal{F}]^{n-1}_{\mathsf{W}_T}\right)$. The protocol uses $r \in \omega(n \ln \lambda)$ instances of each remaining subfunctionality $[\mathcal{F}]^{n-1}_{\mathsf{P}_l} \notin I_\times$. 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 $\mathcal{F}^n_{\mathsf{BC}}$ follows trivially from a $\mathcal{F}^n_{\mathsf{COM}}$ by immediately unveiling the commitment. Furthermore, we showed in Theorem 7, that $\mathcal{F}^n_{\mathsf{COM}}$ follows from $\mathcal{F}^{n-1}_{\mathsf{COM}}$. We show in Appendix A.2 that $\mathcal{F}^{n-1}_{\mathsf{COM}}$ follows from $\mathcal{F}^{n-1}_{\mathsf{FCOT}}$. Hence, our construction works in the $\{\mathcal{F}^{n-1}_{\mathsf{FCOT}}, \mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model; we still use the terms $\mathcal{F}^{n-1}_{\mathsf{COM}}$ and $\mathcal{F}^n_{\mathsf{CG}}$ 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* mechanism 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^λ.

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 and threshold shares, respectively.

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 S, S creates a mask $w \in \{0,1\}$ and uses the additive T-sharing scheme to create shares of $(m_0 \oplus w)$ and $(m_1 \oplus w)$ called $(\mu_j^0)_{j=1..T}$ and $(\mu_j^1)_{j=1..T}$, respectively. S then uses the r-sharing to create shares $(\alpha_{j,i}^b)_{i=1..r}$ for each share μ_j^b with a $(2r/3, r)$-threshold secret sharing scheme. S inputs $(\texttt{messages}, \alpha_{j,i}^0, \alpha_{j,i}^1)$ into the i-th instance of $[\mathcal{F}]_{\mathsf{P}_j}^{n-1}$ for all $i \in [r]$ and $j \in [T]$ and commits globally to w by sending w to $\mathcal{F}_{\mathsf{COM}}^n$ and to each share by sending $\alpha_{j,i}^0$ resp. $\alpha_{j,i}^1$ into $\mathcal{F}_{\mathsf{COM}}^n$ for each $i \in [r]$ and $j \in [T]$.
3. On input $(\texttt{choice}, c)$ with $c \in \{0,1\}$ from $\mathcal{Z}$ to R, R sends $(\texttt{choice}, c)$ to all remaining subfunctionalities $[\mathcal{F}]_{\mathsf{P}_l}^{n-1}$ for all $l \in [T]$. Additionally, R sends c to $\mathcal{F}_{\mathsf{COM}}^n$.
4. If subfunctionality $[\mathcal{F}]_{\mathsf{P}_l}^{n-1} \notin I_\times$ is aborted, the current iteration ends and $[\mathcal{F}]_{\mathsf{P}_l}^{n-1}$ is added to $I_\times$. All parties continue at 1.
5. When a party P has received $(\texttt{receipt messages})$ resp. $(\texttt{receipt choice})$ from all r instances of all type-1 FCOT-subfunctionalities that include P, 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 $\mathcal{S}$, R reconstructs the μ_j^c from the r shares $\alpha_{j,i}^c$. If this succeeds the receiver sends $(\texttt{receipt message})$ to $\mathcal{F}_{\mathsf{BC}}^n$. Otherwise—if reconstruction of a share μ_j^c for $j \in [T]$ fails—the receiver accuses the sender of being a disruptor by sending $(\texttt{invalid}, c, j)$ to $\mathcal{F}_{\mathsf{BC}}^n$.
7. When receiving $(\texttt{invalid}, c, j)$ from R via $\mathcal{F}_{\mathsf{BC}}^n$, S inputs $(\texttt{unveil message}, c)$ to all r subfunctionalities used for $(\alpha_{j,i}^c)_{i=1..r}$ and sends $\texttt{unveil}$ to the instances of $\mathcal{F}_{\mathsf{COM}}^n$ and finishes the protocol execution.
   All parties check whether the sender's FCOT inputs are equal to its global commitments and then biseparate the CG on all subsets of parties that correspond to inconsistent subfunctionalities by declaring conflicts with either the malicious sender or the wronly accusing receiver and its resp. loyalists.
   When receiving $(\texttt{receipt message})$ from R via $\mathcal{F}_{\mathsf{BC}}^n$, the sender S opens the global Commitment to w.
8. R reconstructs $m_c \oplus w$ by combining μ_j^c. Finally, from the unveil of the global Commitment, R reconstructs m_c by applying w to the result.
9. Each W_i and 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 $\mathsf{P} \in [n] \setminus \{\mathsf{S}\}$ 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.
10. Upon receiving $(\texttt{unveil message}, b)$ from $\mathcal{Z}$ to S, S sends $(\texttt{unveil})$ to all $\mathcal{F}_{\mathsf{COM}}^n$. Upon receiving their shares, the parties output $(\texttt{unveil message}, b, m_b)$.
11. Upon receiving $(\texttt{unveil choice})$ from $\mathcal{Z}$ to R, R inputs $(\texttt{unveil choice})$ into all FCOT-functionalities and $(\texttt{unveil})$ into $\mathcal{F}_{\mathsf{COM}}^n$ to unveil c. The other parties first check consistency of all unveiled choice bits. If not consistent, all honest parties declare conflict with R. If the choice bits are consistent c', then the other parties check consistency between the unveiled choice bits of the FCOT-instances 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. If no biseparation has been reached, then each (honest) party outputs the globally unveiled choice bit c as local output.

Now, we give a description of the simulator.

On input $(\texttt{commit}, w)$ from corrupted S to $\mathcal{F}_{\mathsf{COM}}^n$, the simulator lets S input $(\texttt{commit}, w)$ into the simulated $\mathcal{F}_{\mathsf{COM}}^n$.

On input $(\texttt{commit}, \alpha_{j,i}^b)$ from corrupted S to $\mathcal{F}_{\mathsf{COM}}^n$, the simulator lets S input $(\texttt{commit}, (\alpha_{j,i}^b, b, i, j))$ into the simulated $\mathcal{F}_{\mathsf{COM}}^n$.

On input $(\texttt{messages}, \alpha^0_{j,i}, \alpha^1_{j,i})$ from corrupted S to the i-th instance of $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$, once message $(\texttt{commit}, w)$ from S to $\mathcal{F}^n_{\mathsf{COM}}$ has been received, the simulator applies w to both inputs and lets S forward the result to the simulated $[\mathcal{F}]^{n-1}_{\mathsf{P}_j}$.

On input $(\texttt{messages}, \alpha^0_{j,i}, \alpha^1_{j,i})$ from corrupted S to all instances of all remaining FCOT-functionalities, once message $(\texttt{commit}, w)$ from S to $\mathcal{F}^n_{\mathsf{COM}}$ has been received, the simulator computes m_0 and m_1 by combining the obtained shares and applying w on the result. Then the simulator lets S input $(\texttt{messages}, m_0, m_1)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$.

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

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

On output $(\texttt{receipt transfer}, \perp)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$, the simulator gives local input to simulated parties as follows:
If the receiver is malicious, then $\mathcal{S}$ learns m_c from the ideal $\mathcal{F}^n_{\mathsf{FCOT}}$ in the name of R through $(\texttt{receipt transfer}, m_c)$. Otherwise, the simulator gives the uncorrupted R local input $(\texttt{choice}, 0)$. If the sender is uncorrupted, the simulator gives 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 $(\texttt{unveil})$ from corrupted S to the instance of $\mathcal{F}^n_{\mathsf{COM}}$ used to commit to $\alpha^b_{j,i}$, the simulator lets S input $(\texttt{unveil})$ into the respective simulation of $\mathcal{F}^n_{\mathsf{COM}}$.

On input $(\texttt{unveil})$ from corrupted S to the instance of $\mathcal{F}^n_{\mathsf{COM}}$ used to commit to w, the simulator lets S input $(\texttt{unveil})$ into the respective simulation of $\mathcal{F}^n_{\mathsf{COM}}$.

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

On output $(\texttt{unveil message}, b, m_b)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$, 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 equivocate some of the FCOTs and COMs. Therefore, the simulator fabricates additive shares μ^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 equivocate 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 equivocate.
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 equivocate. Furthermore, the simulator knows exactly which shares can be equivocated since every share that the environment obtains comes from the simulator.

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

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

On input $\big(\texttt{abort}, C'\big)$ for $[\mathcal{F}]^{n-1}_{\mathsf{P}_l}$, the simulator aborts $[\mathcal{F}]^{n-1}_{\mathsf{P}_l}$ with $\big(\texttt{abort}, C'\big)$.

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

Also, if an adversarial sender wanted to prepare its share commitments in order to equivocate 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 the $r/3$ altered shares is controlled is bounded by

$$p = \prod_{i=0}^{r/10n-1} \frac{r - r/3 - i}{r - i} \leq \prod_{i=0}^{r/10n-1} 2/3 \leq (2/3)^{r/10n} \tag{4}$$

which is negligible because $r/10n \in \omega(\ln \lambda)$. □

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 S tries to unveil a different value than its original input, it would have to either change the Commitment on the mask w or deviate in more than $r/3$ shares of any one subfunctionality. The former is not possible due to the binding property of $\mathcal{F}^n_{\mathsf{COM}}$. For the latter, note that for each subfunctionality, a party can probe $r/10n$ shares. Thus, 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 3, it follows that at least for two $[\mathcal{F}]^{n-1}_{\mathsf{P}_l} \notin I_\times$, all r instances must be unveiled. Also, R unveils the commitments to its received shares. If R tries to learn both messages, R to have input $(1-c)$ into all instances of at least two FCOT-subfunctionalities; in that case, R cannot learn m_c, since $\left(\mu^c_j\right)_{j \in \{T\}}$ is an additive sharing and thus requires all shares for reconstruction.

We now have proven all the steps depicted in Fig. 3, which leads us to the following conclusion.

Corollary 4 (SFE expansion). *Let n be the number of parties of which at most $0 \leq t \leq (n-3)$ are malicious. The functionality $\mathcal{F}^n_{\mathsf{SFE}}$ can be UC-realized in the $\left\{\mathcal{F}^{n-1}_{\mathsf{SFE}}, \mathcal{F}^n_{\mathsf{BC}}\right\}$-hybrid model:*

$$\left\{\mathcal{F}^{n-1}_{\mathsf{SFE}}, \mathcal{F}^n_{\mathsf{BC}}\right\} \rightsquigarrow \mathcal{F}^n_{\mathsf{SFE}}$$

5 Expanding the result

The result from Corollary 4 allows to expand $(n-1)$-party SFE to n-party SFE, thus proving an upper bound on the minimum cardinality assuming that $t \leq (n-3)$. With earlier results of Rabin and Ben-Or [RB89] and Beaver [Bea90] investigating the honest majority setting, this leaves open the question if we can do better for smaller t between $n/2$ and $n-3$.

In this section, we answer this question positively by extending the results by induction. For that, we additionally require a *multicast*[8] functionality were the recipient set can be chosen by the sender. This allows us to tighten our result for any $t \leq n-3$: As we have shown, $\mathcal{F}^{n-2}_{\mathsf{SFE}}$ and $\mathcal{F}^n_{\mathsf{BC}}$ realize $\mathcal{F}^{n-1}_{\mathsf{SFE}}$ for $t \leq n-3$. If we assume that $t \leq n-4$, then we have that $\mathcal{F}^{n-3}_{\mathsf{SFE}}$ and $\mathcal{F}^{n-1}_{\mathsf{BC}}$ realizes $\mathcal{F}^{n-2}_{\mathsf{SFE}}$. By induction we get $\mathcal{F}^{t+2}_{\mathsf{SFE}}$ and all $\mathcal{F}^i_{\mathsf{BC}}$ for $i \in \{t+3, \ldots, n\}$ realize $\mathcal{F}^n_{\mathsf{SFE}}$.

However, we stress that this implies a limitation regarding the number of induction steps and hence, the number of parties n due to the additional runtime.

We have to do at most $n/2$ inductions, which results in a runtime of

$$(n)^c \cdot (n-1)^c \cdots (n/2)^c = (n!/(n/2-1)!)^c \tag{5}$$

for $c \in \mathrm{poly}(\lambda)$. This implies that we have to choose $n \in \mathcal{O}(\ln \lambda / \ln \ln \lambda)$. This bound depends on our SFE-protocol; using a more efficient one would relax this restriction. For example, with polylogarithmic $n \in \mathrm{polylog}(\lambda)$ and elementary expansion runtime $L \in \mathrm{polylog}(n)$ we get $\mathrm{o}(n^\varepsilon)$ inductions for some positive $\varepsilon < 1$. However, we conjecture that we cannot arbitrarily relax this

[8] Here, a global broadcast with suitable masking technique could provide such a multicast. We leave this to future work.

restriction, due to the following observation that links cheater extraction of a biseparated Conflict Graph to an NP-hard problem:

Recall the generic IA conditions from Section 3 state that IA is possible if the induced CG is biseparated. However, this is only a possibility statement; in general, a t-settled CG does not suffice because it is generally hard to extract the settled set X from the CG. In fact, finding the intersection of all MVCs of size at most t is an abstract graph-theoretical problem on its own. To the best of our knowledge, this problem has not been investigate in the literature; we hence study it as the Intersecting Minimal Vertex Cover (IMVC) problem. Here, we show that for arbitrary graphs (not necessarily CGs) the IMVC problem is NP-hard by giving a Cook-reduction to the standard MinVC problem. The Decisional Minimal Vertex Cover problem [Kar72] states the hardness of deciding whether a given graph G has a Vertex Cover of size at most k. We use the abbreviation MinVC for Minimum VCs and MVC for minimal VCs.

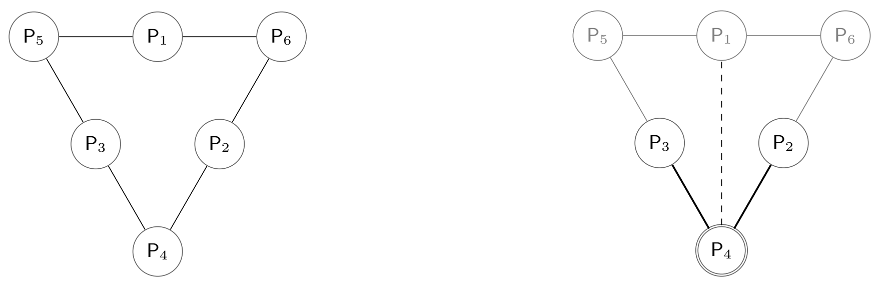
Algorithm 1 hasMinVC($G = ([n], E), k$)

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

In the following, we provide a Cook-reduction of IMVC problem to the decisional MinVC problem; it requires at most n^2 calls to an oracle for the IMVC problem findIMVC to solve the decisional MinVC problem. This yields the somewhat surprising insight that either the special class of Conflict Graphs yields exclusively easy MinVC instances which can be solved efficiently, or finding the settled set of an arbitrary CG is actually NP-hard.

To analyze why Algorithm 1 yields a valid reduction we have to check its runtime and its correctness. In each outer while-loop, the candidate set X grows by at least one party. Hence, the outer while-loop is executed at most n times. Verifying that a set of nodes corresponds to a VC can be done in time linear in n. The oracle findIMVC is called at most once per inner loop-iteration, i.e. at most n^2 times. Finally, removing isolated nodes comes down to checking if any nodes without neighbors exist and removing them; this can be done in time $\mathcal{O}(n)$.

Analyzing the correctness is more tricky. Intuitively, the algorithm accumulates a candidate vertex set X that always remains a subset of some MinVC but steadily increases. Our main focus is on the two lines 9 and 11 where the new vertices are chosen that are added to the candidate set X. In both cases we have to ensure that the nodes of X' are in a common MinVC with the already chosen X. If line 9 gets executed, then $X' := \text{findIMVC}\big(G', t\big)$ is chosen as vertices that are in all remaining MVC. Hence, adding these vertices to X still yields a subset of some MinVC of G. If line 11 gets executed, no IMVC exists in G'. The reduction adds the lexicographically first node (this could as well be a random node without any loss of correctness). The reason that any vertex can be added to X while maintaining a subset of a MinVC is that each vertex is contained



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

(b) Deduction from a given CG. From P_1's view, P_4 has to be corrupted if $t = 3$.

Fig. 6: An example of a graph that is not biseparated but produces a biseparated deduced graph. Thick lines are relevant for the respective property.

in at least one MinVC of G'. If there was one vertex without a corresponding MinVC, then all its neighbors—whose existence is guaranteed by removeIsolatedNodes—must be in all MVCs, in contradiction to the case condition.
The loop invariant that X is a subset of some MinVC of G is maintained in both cases. Finally, we get either that X becomes a VC, thus the MinVC answer is true, or the size of X exceeds k, then the size of each MinVC of G must also exceed k, the answer is false.

While the reduction shown above showcases NP-hardness of IMVC over general graphs, we stress that even if the hardness remains on CGs, where we know that no edges between one partition (the honest parties) can ever exist, we can solve the problem for sufficiently small n: For $n \in \mathcal{O}(\ln \lambda) \vee n - t \in \mathcal{O}(1)$, we introduce an algorithm called DeduceCG, which can deduce a biseparated graph from any t-settled graph in polynomial-time by intersecting all VCs of size at most t. Hence, for $n \in \mathcal{O}(\ln \lambda) \vee n - t \in \mathcal{O}(1)$, a t-settled induced CG already implies the ability to abort.

Algorithm 2 DeduceCG(G, t)

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

Informally, the algorithm adds new edges whenever a party's view can only be explained by a certain other party being malicious. The algorithm sequentially takes on the role of each party $\mathsf{P} \in [n]$ in the induced CG $G = ([n], E)$. Take for example the graph in Fig. 6a, which is not 3-settled since the vertex covers $\{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3\}$ and $\{\mathsf{P}_4, \mathsf{P}_5, \mathsf{P}_6\}$ do not contain any common party. The algorithm starts by taking the viewpoint of P_1; in this role, any conflict edge of the form $\{\mathsf{P}_1, \mathsf{P}'\}$ implies that P' is a disruptor, whereas conflicts between two other parties, e.g. $\{\mathsf{P}_3, \mathsf{P}_5\}$, leaves some uncertainty with respect to the corruptions. Hence, the algorithm computes all explanations subject to the fact that the neighbors of P_1 are corrupted, and which are of size $\leq t$. To those, it declares new conflicts, despite the fact that these two parties may have never even interacted with one another. The algorithm then repeats this step for each party $\mathsf{P} \in [n]$.

More formally, let $\mathrm{N}(\mathsf{P})$ be the neighbors of P in G, that is, all parties P' for which $\{\mathsf{P}, \mathsf{P}'\} \in E$. The algorithm checks, if the rest of the graph on $[n] \setminus (\{\mathsf{P}\} \cup \mathrm{N}(\mathsf{P}))$ is $(t - |\mathrm{N}(\mathsf{P})|)$-settled and appends conflicts between P and all parties in the settled set of this subgraph.
For the induced Conflict Graph from Fig. 6a with $t = 3$, this would mean that DeduceCG would take on the role of 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 P_4, DeduceCG adds an edge from P_1 to P_4 in Fig. 6b.

In an information-theoretic sense the DeduceCG algorithm *completes* an induces CG. The deduced CG contains all conflicts, even those which might be hard to generate by finding the settled set of the induced CG and biseparating it from its complement. This is exactly the reason why DeduceCG is at least as hard as finding the IMVC of the CG.

Many of the results we achieve from the Conflict Graph exploit the guarantee that each edge contains at least one disruptor. While this is obviously the case for edges which are caused by subfunctionality abort and has to be ensured during protocol-specific accusations, this is not so obvious for edges which were added by the DeduceCG algorithm. However, we stress that this is the case.

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

Proof. We prove our claim by contradiction. Let $\{\mathsf{P}_i, \mathsf{P}_j\}$ be a an edge in E^*. The edge can have only two possible origins. Either it was part of the original Conflict Graph, that is, $\{\mathsf{P}_i, \mathsf{P}_j\} \in E$; this would be a contradiction to our assumption regarding $G = \{[n], E\}$. Or the edge was added during DeduceCG. From the conditions that have to be fulfilled according to Algorithm 2 in order for an edge to be added, we can infer a lot of information. Without loss of generality, we assume that the edge was added while the view of P_i was investigated and that P_i is honest; that is, DeduceCG deduced that in the view of P_i, P_j must be malicious.

Let $G' = \{[n] \setminus \{\mathsf{P}_i \cup \mathrm{N}(\mathsf{P}_i)\}, E\}$ be the conflict graph that excludes P_i and its neighbors $\mathrm{N}(\mathsf{P}_i)$. By requirement from Algorithm 2, G' is t'-settled for $t' = (t - |\mathrm{N}(\mathsf{P}_i)|)$ and furthermore, all explanations of t' corrupted parties contain P_j as corrupted party. This implies for the original conflict graph containing all parties, that P_j is also selected in the cover of size t in $[n]$. P_j also has a set of direct neighbors $\mathrm{N}(\mathsf{P}_j)$ in G; if we now assume $\{\mathsf{P}_i, \mathsf{P}_j\}$ to be an honest-honest edge, thus if P_j is honest, by correctness of G all neighbors $\mathrm{N}(\mathsf{P}_j)$ are corrupted.

Let C be the set of corrupted parties. By requirement, it holds that $|C| \le t$. If P_i is honest, then it holds that $|C| - |\mathrm{N}(\mathsf{P}_i)| \le t'$. Thus, there is a valid explanation of size t' that contains all corrupted parties in $C \setminus \mathrm{N}(\mathsf{P}_i)$. If P_j is honest, C contains $\mathrm{N}(\mathsf{P}_j)$ entirely. Thus, there is an explanation that *excludes* P_j, meaning that the subgraph G' is not t'-settled and hence, the DeduceCG-algorithm will not add the honest-honest edge $\{\mathsf{P}_i, \mathsf{P}_j\}$. □

Note that our SFE-expansion (Section 4) doesn't suffer from the logarithmic limitation of n because we actually don't need the DeduceCG algorithm. Our construction only uses hybrid functionalities of size $(n - 1)$; the only way for an adversary to force an abort is to abort many subfunctionalities. Fortunately, Lemmas 9 to 11 already yield a *biseparated* CG as a result of too many subfunctionality aborts (as opposed to t-settledness). Therefore, when an abort is forced by the adversary, it is also possible to identify disruptors because of the biseparation of the induced CG.

Here, we want to point out an interesting observation: when the setup, that is, the size of the hybrid functionality, is large, e.g. $(n - 1)$, then the deduced CG does not yield an advantage over the induced CG due to the aforementioned biseparation guarantee of ours abort lemmas. However, if the setup is small, e.g. 2, then (statistically secure) protocols for SFE might become impossible, as shown by [IOS12] without broadcast. This begs the question if the minimal complete cardinality (minimal setup size) is different when a functionality for the induced resp. deduced graph is provided. There is no difference for small n since the deduction can be efficiently computed. However, for superlogarithmic n this is not clear and we pose it as an open question for further research.

Lastly, if we use induction to extend our SFE-expansion (comp. Corollary 2) we have a slightly sublogarithmic bound on n again; this bound stems from the efficiency of the elementary SFE-expansion and can only be remedied by improving the efficiency of the SFE-expansion protocol to sublinear in n. We also pose this as an open question for further research.

6 Summary and Outlook

Summary. We formalized an intuitive 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 and t-settledness, which we hope will contribute to further research in the field of Identifiable Abort. In particular, we have shown that any protocol with Identifiable Abort can be augmented to yield a biseparated CG upon abort, and that the abort parties are fixed as soon as the CG becomes t-settled. We elaborated that for a sufficiently small number of parties n, there exist an efficient deduction rule to biseparate any t-settled CG, which leads to a conjectured NP-hardness of Identifiable Abort.

We extended Oblivious Transfer to yield a helpful tool in the IA-setting. Our variant, Fully Committed Oblivious Transfer (FCOT), is provably as powerful as Secure Function Evaluation in the setting of IA, but much 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 adversaries without any runtime restrictions, as long as at least three parties are honest. This SFE-expansion yields an upper bound on the minimal complete cardinality $k^*(t)$ in the sense of [FGM+01]: given a broadcast channel, we have shown that $k^*(t) \leq (t+2)$ for $n \in \mathcal{O}(\ln \lambda / \ln \ln \lambda)$. 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 introduced a new methodology to analyze protocols with Identifiable Abort, new open questions naturally arise. The first open problem is regarding the efficiency of our protocols. One other direction we believe to be worth investigating is to drop the requirement of the broadcast channel. Another open issue is to tighten our upper bound on the minimal complete setup size k^*; here one would need to provide abort-lemmas similar to Lemmas 9 to 11 for subfunctionalities of cardinality $n-2$ or less.

A different research direction is to clarify the time complexity of computing the settled set X as defined in Definition 2 in a CG that is not biseparated. 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, we know that for a logarithmic number of parties, t-settled Conflict Graphs can be efficiently transformed into biseparated Conflict Graphs, from which the identification is trivially possible. In Section 5, we already provided an algorithm DeduceCG to compute the deduced Conflict Graph from a given induced Conflict Graph. The logarithmic bound on the number of parties comes from the requirement to compute X, which contains all MVCs of size $\leq t$. While we know that this problem is NP-hard for general graphs, we do not know yet if the class of Conflict Graphs provide some property that can be used for a more efficient solution of this problem.

References

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In pages 99–108, 1996.

[AL07] Y. Aumann and Y. Lindell. Security against covert adversaries: efficient protocols for realistic adversaries. In pages 137–156, 2007.

[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.

[BMM+20] N.-P. Brandt, S. Maier, T. Müller, and J. Müller-Quade. Constructing secure multi-party computation with identifiable abort. Cryptology ePrint Archive, Report 2020/153, 2020. https://eprint.iacr.org/2020/153.

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

[BOS$^+$20] C. Baum, E. Orsini, P. Scholl, and E. Soria-Vazquez. Efficient constant-round mpc with identifiable abort and public verifiability. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 562–592, Cham. Springer International Publishing, 2020.

[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.

[CL14] R. Cohen and Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In pages 466–485, 2014.

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

[CLO$^+$02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In pages 494–503, 2002.

[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.

[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.

[GM82] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In pages 365–377, 1982.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. 28(2):270–299, 1984.

[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.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[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.

[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.

[SSY21] M. Simkin, L. Siniscalchi, and S. Yakoubov. On sufficient oracles for secure computation with identifiable abort. Cryptology ePrint Archive, Report 2021/151, 2021. https://eprint.iacr.org/2021/151.

A Relations between Functionalities

In this section, we investigate which of the aforementioned functionalities imply each other. Mainly, we show that $\mathcal{F}^n_{\mathsf{FCOT}}$ and $\mathcal{F}^n_{\mathsf{SFE}}$ are equivalent in the IA-setting, and that $\mathcal{F}^n_{\mathsf{FCOT}}$ suffices to construct $\mathcal{F}^n_{\mathsf{COM}}$.

A.1 SFE-Completeness of FCOT

In this section, we provide the proofs 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 13 (SFE $\rightsquigarrow$ FCOT). *Let n be any number of parties. There is a protocol π^n_{FCOT} in the $\{\mathcal{F}^n_{\mathsf{SFE}}\}$-hybrid model that securely UC-realizes $\mathcal{F}^n_{\mathsf{FCOT}}$.*

Proof. We denote the set of all parties by $[n] = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, \ldots, \mathsf{W}_{n-2}\}$. 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 use several secret sharings to share its inputs. We use a secret sharing to create n shares for every input. In fact, we do this r times, ending up in a total of $r \cdot n$ shares for every input.

Set the number of shares to $r = n \ln^2 \lambda$. For $i \in [r]$, the sharings $\bigoplus_{j=1}^n \mu^0_{j,i} =: m_0$ and $\bigoplus_{j=1}^n \mu^1_{j,i} =: m_1$ 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 receiver R obtains sufficiently many shares to reconstruct one message, but not enough to reconstruct both. The witnesses 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. To achieve this, all the witnesses chose one of the n shares for all of r sharings randomly.

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

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

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

$$\begin{aligned}
x_{\mathsf{S}} &\coloneqq \Big(\big(\mu^0_{j,i}, \mu^1_{j,i}\big)_{j=1..n}, \nu^i_{\mathsf{S}} \Big)_{i=1..r} & x_l &\coloneqq \big(\nu^i_l\big)_{i=1..r} \\
y_{\mathsf{S}} &\coloneqq \Big(\gamma_{\nu^i_{\mathsf{S}}} \Big)_{i=1..r} & y_l &\coloneqq \Big(\mu^{0,i}_{\nu^i_l}, \mu^{1,i}_{\nu^i_l}, \gamma_{\nu^i_l} \Big)_{i=1..r} \\
x_{\mathsf{R}} &\coloneqq \Big(\big(\gamma_{j,i}\big)_{j=1..n}, \nu^i_{\mathsf{R}} \Big)_{i=1..r} & u^b_{\mathsf{S}} &\coloneqq \big(\mu^b_{j,i}\big)_{j=1..n, i=1..r} \\
& & \varDelta &\coloneqq (\varDelta_{\mathsf{S}}, \varDelta_{\mathsf{R}}) \\
y_{\mathsf{R}} &\coloneqq \Big(m_c, \Big(\mu^{0,i}_{\nu^i_{\mathsf{R}}}, \mu^{1,i}_{\nu^i_{\mathsf{R}}} \Big)_{i=1..r} \Big) & v_{\mathsf{R}} &\coloneqq \big(\gamma_{j,i}\big)_{j=1..n, i=1..r}
\end{aligned}$$

The common output $\varDelta_{\mathsf{S}}$ 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 $\varDelta_{\mathsf{S}}$ is 0. Analogously, $\varDelta_{\mathsf{R}}$ 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 ε. Additionally, each party implicitly obtains a unary representation of the security parameter 1^λ.

The protocol looks as follows:

On input $(\texttt{local start})$ from $\mathcal{Z}$ to W_l, the witnesses choose their shares by drawing r indices $\left(\nu^i_l\right)_{i\in[r]} \overset{\$}{\leftarrow} \mathbb{Z}^r_n$, to determine which share of each sharing of the other parties' inputs will be obtained by W_l.

On input $(\texttt{messages}, m_0, m_1)$ from $\mathcal{Z}$ to S, 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, 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 S obtains. S inputs x_{S} into $\mathcal{F}^n_{\mathsf{SFE}}[f]$.

On input $(\texttt{choice}, c)$ from $\mathcal{Z}$ to R, 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, 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 R obtains. R inputs x_{R} into $\mathcal{F}^n_{\mathsf{SFE}}[f]$.

On output $\Delta_{\mathsf{S}} = 0$ or $\Delta_{\mathsf{R}} = 0$ from $\mathcal{F}^n_{\mathsf{SFE}}[f]$ to P, where $\mathsf{P} \in [n]$, the party P aborts with output $\left(\texttt{abort}, C'\right)$, where $\mathsf{S} \in C' \iff \Delta_{\mathsf{S}} = 0$ and $\mathsf{R} \in C' \iff \Delta_{\mathsf{R}} = 0$.

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

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

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

On output u^b_{S} from $\mathcal{F}^n_{\mathsf{SFE}}\left[g^b_{\mathsf{S}}\right]$ to P, where $\mathsf{P} \in [n] \setminus \mathsf{S}$, 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, P outputs $(\texttt{unveil message}, b, m_b)$. Otherwise, P outputs $(\texttt{abort}, \mathsf{S})$.

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

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

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

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

On output $(\texttt{unveil message}, b, m_b)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$, the simulator fabricates input u^b_{S} for the uncorrupted simulated S to send as input into $\mathcal{F}^n_{\mathsf{SFE}}\left[g^b_{\mathsf{S}}\right]$. The simulator knows the indices of all shares of μ^0 and μ^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 μ^0 and μ^1 encode 0. Denote the manipulated sharings by ${\mu'}^0$ and ${\mu'}^1$; they encode m_0 resp. m_1. Finally, the simulator lets S input ${u'_\mathsf{S}}^b$ (computed from ${\mu'}^b$) into $\mathcal{F}^n_{\mathsf{SFE}}\big[g^b_\mathsf{S}\big]$. 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 $\mathcal{F}^n_{\mathsf{FCOT}}$, the simulator fabricates input u_R for the uncorrupted simulated R to input into $\mathcal{F}^n_{\mathsf{SFE}}[g_\mathsf{R}]$. 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 $\mathcal{F}^n_{\mathsf{SFE}}[\cdot]$, the simulator aborts $\mathcal{F}^n_{\mathsf{SFE}}[\cdot]$ as well as $\mathcal{F}^n_{\mathsf{FCOT}}$ 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 S to $\mathcal{F}^n_{\mathsf{SFE}}[f]$, the simulator lets S input $(\texttt{messages}, m_0, m_1)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$, 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 $\mathcal{F}^n_{\mathsf{SFE}}[f]$.

On input $(\texttt{input}, x_\mathsf{R})$ from corrupted R to $\mathcal{F}^n_{\mathsf{SFE}}[f]$, the simulator lets R input $(\texttt{choice}, c)$ into $\mathcal{F}^n_{\mathsf{FCOT}}$, 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 $\mathcal{F}^n_{\mathsf{SFE}}[f]$.

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

On input $(\texttt{input}, u_\mathsf{R})$ from corrupted R to $\mathcal{F}^n_{\mathsf{SFE}}[g_\mathsf{R}]$, the simulator checks consistency of the sharings in v_R with the sharings in x_R. If they are consistent, then the simulator lets R input $(\texttt{unveil choice})$ into $\mathcal{F}^n_{\mathsf{FCOT}}$, otherwise the simulator aborts $\mathcal{F}^n_{\mathsf{FCOT}}$ 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 $\mathcal{F}^n_{\mathsf{SFE}}[f]$, then, upon termination of $\mathcal{F}^n_{\mathsf{SFE}}[f]$, all honest parties certainly abort with output $(\texttt{abort}, \mathsf{S})$ due to the common output Δ. Also, if inconsistent sharings for m_0 are unveiled in $\mathcal{F}^n_{\mathsf{SFE}}[g_\mathsf{S}]$, 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_P be a boolean random variable and $H^i_\mathsf{P} = 1$ be the event that the i-th share index of (honest) party P matches the index of the maliciously altered share, else $H^i_\mathsf{P} = 0$. We get the probability $\Pr\big[H^i_\mathsf{P} \bigm| Z \geq 1\big] \geq 1/n$. Let $N_\mathsf{P} \coloneqq \bigvee_{i=1}^r H^i_\mathsf{P} = \neg\bigwedge_{i=1}^r \neg H^i_\mathsf{P}$ be the boolean random variable that describes whether P notices an alteration in any of the r sharings. We provide an upper bound for the probability $N_{\mathsf{P}} = 1$ by

$$\begin{aligned}
\Pr[N_{\mathsf{P}} \mid Z \geq 1] &= 1 - \Pr[\neg N_{\mathsf{P}} \mid Z \geq 1] \\
&= 1 - \Pr\left[\bigwedge_{i=1}^{r} \neg H_{\mathsf{P}}^{i} \;\middle|\; Z \geq 1\right] \\
&= 1 - \prod_{i=1}^{r} \Pr\left[\neg H_{\mathsf{P}}^{i} \mid Z \geq 1\right] \\
&\overset{(6.1)}{\geq} 1 - \prod_{i=1}^{r} (1 - 1/n) \\
&= 1 - (1 - 1/n)^{r}
\end{aligned} \tag{6}$$

where (6.1) uses the fact that the different H_{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 $\left(1 - (1 - 1/n)^{r}\right)^{h}$ which can be bounded by $p > \left(1 - (1 - 1/n)^{r}\right)^{n}$. We see that

$$\begin{aligned}
p &> \left(1 - (1 - 1/n)^{r}\right)^{n} \\
&= \left(1 - (1 - 1/n)^{n \ln^{2} \lambda}\right)^{n} \\
&> \left(1 - (2/e)^{\ln^{2} \lambda}\right)^{n} \\
&> (2e)^{-n (2/e)^{\ln^{2} \lambda}} \\
&\in \exp(-\operatorname{negl}(\lambda))
\end{aligned} \tag{7}$$

which is overwhelming in λ for any $n \in \operatorname{poly}(\lambda)$. Although our choice for r is probably not tight, we already see that the protocol is at most linear in n and superlogarithmic in λ.

□

Before showing the other implication we first recall the so-called IPS-compiler [IPS08].

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

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

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

The parameters for the secret sharing scheme are chosen such that on the one hand side the probability for unnoticed deviation from the inner protocol is negligible while on the other hand no information from the watched servers allows reconstruction of the clients actual inputs; for further details we refer to [IPS08].

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

While Cohen and Lindell [CL14] already obtained MPC with IA by making non-black-box use of the GMW-compiler [GMW87], we claim that the IPS compiler can also be modified so as to yield IA: By replacing two-party OT $\mathcal{F}^2_{\mathsf{OT}}$ with Fully Committed Oblivious Transfer $\mathcal{F}^n_{\mathsf{FCOT}}$, any client that detects the misbehavior can post hoc request the unveiling of all communication regarding the affected server and the corresponding one-time pads such that all parties can retrace the inner protocol and identify which party made inputs to the affected server that do not match the value on the watchlist broadcast channel:

Lemma 14 (FCOT $\rightsquigarrow$ SFE). *Let n be any number of parties. There is a protocol π^n_{SFE} in the $\{\mathcal{F}^n_{\mathsf{FCOT}}\}$-hybrid model that securely UC-realizes $\mathcal{F}^n_{\mathsf{SFE}}$.*

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

We use the IPS-compiler [IPS08] described above, which compiles two protocols Π and ρ into an $\left\{\mathcal{F}^2_{\mathsf{OT}}\right\}$-hybrid protocol Φ. Their result cannot be directly transferred into the setting of IA. However, we slightly modify their protocol in the following ways: (1) All communication in the new inner protocol is processed via FCOTs. (2) We replace $\mathcal{F}^2_{\mathsf{OT}}$-calls with to $\mathcal{F}^n_{\mathsf{FCOT}}$. (3) When a client P would abort in the original protocol [IPS08] due to noticing misbehavior in server i, it instead publicly demands the unveiling of all communication corresponding to server i. After the unveil, it holds that either there is a set of disruptors C' that was identified by this procedure or that refused to unveil their communication,[9] then all honest parties send $(\texttt{conflict}, C')$ to $\mathcal{F}^n_{\mathsf{CG}}$. Or the unveiled messages indicate no malicious behavior on that server, in which case all honest parties send $(\texttt{conflict}, \mathsf{P})$ to $\mathcal{F}^n_{\mathsf{CG}}$. Note that Item (1) is without loss of generality as OT (hence also FCOT) suffices to set up authenticated channels, but allows unveiling communication upon accusation of misbehavior. Item (2) also does not change the behavior of the simulator as the extraction that Ishai, Prabhakaran, and Sahai [IPS08] use for OT is compatible with the definition of FCOT. To prove security we can hence use essentially the same simulator that was also used by Ishai, Prabhakaran, and Sahai [IPS08], which only has to be adjusted to incorporate the new Identifiable Abort criteria from Item (3).

Item (3) only changes behavior with respect to aborts, so if no aborts occur then simulation is exactly the same and the modified simulator learns as much information as the original simulator from [IPS08]. The only adaptations to the simulator are with respect to detected misbehavior of parties: instead of merely forwarding the abort of a single party to the functionality our simulator must provide a set of corrupted parties C' to abort the ideal functionality $\mathcal{F}^n_{\mathsf{SFE}}[f]$, and all simulated parties in the protocol must provide output $\left(\texttt{abort}, C'\right)$. The new protocol ensures this in the following way. The key to the new behavior is a parties message $(\texttt{challenge}, i)$ which a party can use to receive an explanation of the behavior regarding the i-th server. This can either be caused by the simulator directly in the name of an honest party after detecting misbehavior in the i-th server, or by a malicious party who is controlled by the environment. Yet both scenarios are handled equivalently by the simulator. The challenge causes all parties to unveil the FCOTs used to distribute their watchlist one-time pads and explain the messages that are related to the i-th instance of the inner protocol (that is, the i-th server). More precisely, we assume that for distributing the watchlist keys a $\binom{n^2\lambda}{\lambda}$-FCOT was used for each party which can be canonically

[9] The synchronous model allows parties to notice when a party denies unveiling.

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. Again, either the challenging party indeed identified a malicious message on the i-th server, then all parties notice this misbehavior and identify the disruptor party P and abort with $(\texttt{abort}, \mathsf{P})$. Or the challenging party lied about receiving a malicious message on the i-th server, which only happens if the calling party is really malicious; in which case all other parties will abort with $\left(\texttt{abort}, \mathsf{P}'\right)$ where P' corresponds to the party that sent challenged server i. Thus the simulator can extract the disruptor and use it to abort the ideal functionality $\mathcal{F}^n_{\mathsf{SFE}}$ in the name of that party. Note that even in the original simulator [IPS08], after a misbehavior has been detected by the simulator it corrupts that server in the inner protocol; hence our induced changes leak no additional information to the adversary.

However, we must still ensure that unveiling all inputs of a single server does not violate the privacy of the clients in the outer protocol Π. In the following, we formalize this idea: let n be the number of clients in the outer protocol. In the original paper [IPS08] there are $m \in \Theta\left(n^2\lambda\right)$ servers. Each party gets to select λ watchlists from each party, such that each party can see all in- and outcoming communication of λ servers. In total, at most a fraction of $n\lambda/n^2\lambda = 1/n$ of all servers state is known by any set of parties. Because the used secret sharing requires a constant fraction of shares to reconstruct the original input, no coalition of parties can learn the input of another party. Now, if misbehavior occurs and the state of an additional server is 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.

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

A.2 Global Commitment from Fully Committed Oblivious Transfer

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

Lemma 15. *There is a protocol π^n_{COM} that securely UC-realizes $\mathcal{F}^n_{\mathsf{COM}}$ in the $\{\mathcal{F}^n_{\mathsf{FCOT}}\}$-hybrid model:*

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

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

On input $(\texttt{commit}, m)$ for $m \in \{0,1\}$ from $\mathcal{Z}$ to C, C acts as R in $\mathcal{F}^n_{\mathsf{FCOT}}$ and inputs $(\texttt{choice}, m)$.
On input $(\texttt{receipt commit})$ from $\mathcal{F}^n_{\mathsf{COM}}$ to R_i for $i \in [n-1]$, all receiver for $i \neq 1$ ignore the message. R_1 acts as the sender in $\mathcal{F}^n_{\mathsf{FCOT}}$: it draws one random message $m' \overset{\$}{\leftarrow} \{0,1\}$ and sends $(\texttt{messages}, m', m')$ to $\mathcal{F}^n_{\mathsf{FCOT}}$.
On input $(\texttt{unveil})$ from $\mathcal{Z}$ to C and $(\texttt{receipt transfer}, m_c)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ to C, C sends $(\texttt{unveil choice})$ to $\mathcal{F}^n_{\mathsf{FCOT}}$.
On input $(\texttt{unveil choice}, c)$ from $\mathcal{F}^n_{\mathsf{FCOT}}$ to any receiver R_i for $i \in [n-1]$, R_i outputs $(\texttt{output}, c)$.

The protocol essentially exploit the committed nature of the choice bit provided by $\mathcal{F}_{\mathsf{FCOT}}$.

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

1. If C is corrupted:
   On input $(\texttt{choice}, c)$ from C to $\mathcal{F}^n_{\mathsf{FCOT}}$, $\mathcal{S}$ sends $(\texttt{commit}, c)$ to $\mathcal{F}^n_{\mathsf{COM}}$ in the name of C.
2. If C is honest:
   On input $(\texttt{receipt commit})$ from $\mathcal{F}^n_{\mathsf{COM}}$, $\mathcal{S}$ simulates $\mathcal{F}^n_{\mathsf{FCOT}}$ according to the code of $\mathcal{S}_{\mathrm{FCOT}}$ with arbitrary input.

3. If 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 R_1. Else, $\mathcal{S}$ reports $(\texttt{receipt transfer})$ to $\mathcal{Z}$.
4. If R_i for $i \in [n]$ is honest:
   $\mathcal{S}$ acts according to the protocol of R_i.
5. If C is corrupted:
   On input $(\texttt{unveil choice})$ from R to $\mathcal{F}^n_{\mathsf{FCOT}}$, $\mathcal{S}$ sends c to all R_i for $i \in [n-1]$.
6. If C is honest:
   On input $(\texttt{unveil}, c)$ from $\mathcal{F}^n_{\mathsf{COM}}$, $\mathcal{S}$ reports message $(\texttt{unveil choice}, c)$ to all R_i.

The simulator trivially provides an indistinguishable view:

- Simulation of $\mathcal{F}^n_{\mathsf{FCOT}}$ follows from simulation-based security.
- The only secret is the to-be-committed bit m, as the receivers 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 $\mathcal{F}^n_{\mathsf{FCOT}}$ accordingly and pretend that C used the correct choice bit – which does not have to be known in advance, as $(\texttt{unveil choice}, c)$ is only required *after* $\mathcal{F}^n_{\mathsf{COM}}$ unveiled c.
- Against a corrupted committer, $\mathcal{S}$ learns c via simulation of $\mathcal{F}^n_{\mathsf{FCOT}}$.

Thus, the claim follows. □

B Conflict Graph from Broadcast

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

Lemma 16. *Let n be the number of parties of which at most $0 \leq t < n$ are malicious. There is a protocol π^n_{CG} in the $\{\mathcal{F}^n_{\mathsf{BC}}\}$-hybrid model that securely UC-realizes $\mathcal{F}^n_{\mathsf{CG}}$:*

$$\mathcal{F}^n_{\mathsf{BC}} \rightsquigarrow \mathcal{F}^n_{\mathsf{CG}} \tag{8}$$

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

Initialize. All parties start with a graph $G_i = ([n], E_i)$ with $E_i = \emptyset$.
On input $(\texttt{conflict}, \mathsf{P}_i)$ from $\mathcal{Z}$ to P_j, P_j inputs $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ to $\mathcal{F}^n_{\mathsf{BC}}$.
On input $(\texttt{output}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ from $\mathcal{F}^n_{\mathsf{BC}}$, all parties $\mathsf{P} \in [n]$ add $\{\mathsf{P}_j, \mathsf{P}_i\}$ to E_i.
On input $(\texttt{query})$ from $\mathcal{Z}$ to P_i, P_i outputs it own G_i.

A simulator for this protocol is straightforward:

1. If P is corrupted:
   On input $(\texttt{input}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ from P to $\mathcal{F}^n_{\mathsf{BC}}$, if $\mathsf{P}_j = \mathsf{P}$, $\mathcal{S}$ calls $\mathcal{F}^n_{\mathsf{CG}}$ with input $(\texttt{conflict}, \mathsf{P}_i)$ in the name of P and sends $(\texttt{output}, (\texttt{conflict}, \mathsf{P}_j, \mathsf{P}_i))$ to all other parties in the name of $\mathcal{F}^n_{\mathsf{BC}}$.
2. If P is honest:
   On input $(\texttt{conflict}, \mathsf{P}, \mathsf{P}_i)$ from $\mathcal{F}^n_{\mathsf{CG}}$ to $\mathcal{S}$, $\mathcal{S}$ calls $\mathcal{F}^n_{\mathsf{BC}}$ with input $(\texttt{output}, (\texttt{conflict}, \mathsf{P}, \mathsf{P}_i))$ into $\mathcal{F}^n_{\mathsf{BC}}$ in the name of 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.
- For corrupted parties, $\mathcal{S}$ obtains the input via the simulated $\mathcal{F}^n_{\mathsf{BC}}$. If the broadcasted message was valid, $\mathcal{S}$ inputs this into $\mathcal{F}^n_{\mathsf{CG}}$, thus causing the same behavior as if an honest party had called $\mathcal{F}^n_{\mathsf{CG}}$.
- For honest parties, $\mathcal{S}$ only has to simulate the behavior of $\mathcal{F}^n_{\mathsf{BC}}$.

□