Constructing Secure Multi-Party Computation with Identifiable Abort

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

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

Abstract. Statistical Multi-Party Computation (MPC) protocols based on *two-party* Oblivious Transfer (OT) have one severe drawback: the adversary can abort the protocol without repercussions. [IOZ14] introduced the notion of Identifiable Abort (IA). We extend the work of [FGM+01] and investigate, under which conditions n-party MPC can be constructed from smaller functionalities in the setting of IA. Previous work already contains an impossibility result for two-party functionalities [IOS12] and a universal n-party setup [IOZ14].
We thus investigate setup functionalities of size between 3 and $(n-1)$. In this paper we give novel upper bounds for the sizes of functionalities needed for IA. In particular, we find out, that it is possible to construct n-party MPC with IA from an $(n-1)$-party setup and a broadcast, if at least 3 parties are honest. We achieve our result by using a new and innovative technique called *conflict graph* and its complementary *association graph*, which uses a broadcast channel to model the knowledge of honest parties regarding the identity of malicious parties.

Keywords: Multi-Party Computation · Simulation-Based Security · Identifiable Abort · Conflict Graph

1 Introduction

Minimal Cardinality. [Kil88; IPS08] proved that two-party primitives like OT can be used to construct general n-party MPC secure against arbitrary adversaries, but only in the setting of Anonymous Abort, also known as *security with abort*. Surprisingly, this result does not carry over to the setting of Identifiable Abort [IOS12]. This raises the question, how strong the setup has to be in order to construct n-party MPC with IA. The construction of [IOZ14] requires *Correlated Randomness*, which can be modeled as an n-party functionality and as such poses a relatively strong assumption. We say that the *Correlated Randomness*-functionality is of *size* n. In general, we would like to compose general n-party MPC solely from functionalities of smaller size and a global broadcast. Following [FGM+01], we call the minimal size of a functionality that enables n-party MPC the *minimal complete cardinality* k^* of a primitive.

Identifiable Abort. Ishai, Prabhakaran, and Sahai [IPS08] classify protocols for secure Multi-Party Computation (MPC) into two categories: (1) Protocols which are only secure with an honest majority of parties, and (2) Protocols which are secure against arbitrarily many malicious parties. The former implies some level of trust in the other parties, which can be used for techniques such as *majority votes*. For the latter, it was stated by the authors, that certain security guarantees cannot be made. In particular, an adversary can always abort the computation, without revealing information which allows to identify corrupted parties. We call this property Anonymous Abort (AA). Since this is an undesirable property for the real world, Ishai, Ostrovsky, and Zikas [IOZ14] introduced a compromise between what we would want for real-world protocols and what is technically feasible, which they called Identifiable Abort (IA). It still allows the adversary to abort any protocol at will, but in doing so, the identity of at least one malicious party is revealed to all participants. In our work, we try to find bounds on the minimal complete cardinality in the setting of IA. We refer to ideal functionalities $\mathcal{F}$ with IA as $\mathcal{I}$.

Adversarial Restrictions. In order to circumvent impossibility results or raise the efficiency, security statements often consider only a certain sub-class of adversaries. One common limitation is the computational power of the adversary. Most protocols are only secure against adversaries with *polynomial runtime*, where security can be based on the computational complexity of well-investigated problems. Another common limitation is on the number of parties that an adversary can corrupt. Ben-Or, Goldwasser, and Wigderson [BGW88] showed that if we only consider adversaries who corrupt less than one third of all parties, $k^* = 2$ is the minimal complete cardinality. This result already indicates some relation between the number of corrupted parties and the minimal complete cardinality. Our investigation is focused on this particular relation.

Summary. In this work, we investigate upper and lower bounds of the minimal complete cardinality k^* in the style of [FGM+01], but with Identifiable Abort. As a helpful tool, we formalize the novel concept of the *conflict graph*, which contains information on all conflicts between parties that have occurred during a protocol execution. This tool requires the existence of a global *broadcast* channel, which is of size n, but still a relatively weak assumption. The conflict graph aids the honest parties in identifying a set of malicious parties, when too many conflicts arise.

We use the conflict graph to present a construction that uses $(n-1)$-party Secure Function Evaluation and n-party broadcast to construct n-party SFE, without losing the properties of Identifiable Abort.

1.1 Related Work

The question regarding requirements and cardinalities for hybrid functionalities to achieve general n-party MPC has been investigated for several different scenarios. Such research is motivated by the fact, that hybrid functionalities of smaller

cardinality can be realized more easily using physical assumptions (see e.g. [GIS+10; CK88; Cré97]).

One of the earliest insights in that area comes from the seminal work of Kilian [Kil88] (which was later improved by Ishai, Prabhakaran, and Sahai [IPS08]), which works in the setting of Anonymous Abort (AA). In this setting, the adversary can abort the protocol anonymously – that is, without revealing information regarding corrupted parties – if the state is considered unfavorable. They proved that in this setting, actively secure n-party MPC is possible using only Oblivious Transfer (OT), which is of cardinality 2.

In the setting of Universal Composability by Canetti [Can01], it was shown by Canetti and Fischlin [CF01], that without an additional n-party set-up (like a trusted third party, who creates a common reference string from a given distribution), general MPC cannot be achieved. They even proved that not even (MPC-incomplete) commitments can be securely realized in this setting.

Our work focuses on the setting of Identifiable Abort. This setting was first investigated (under the name *Cheater Identification*) by Ishai, Ostrovsky, and Seyalioglu [IOS12], although only in the honest majority setting. It turned out that even under such seemingly loose restrictions, no composition of functionalities with cardinality 2 (such as OT) exists that leads to n-party MPC with cheater identification and therefore there is no canonical extension of the protocol by Kilian [Kil88], which adds identifiable abort using only OT hybrid functionalities.

The information-theoretic setting of IA was first investigated by Ishai, Ostrovsky, and Zikas [IOZ14]. There, the authors proved that the results of Ishai, Ostrovsky, and Seyalioglu [IOS12] do not hold if there is a correlated randomness setup; which doesn't contradict the statement, since correlated randomness is considered a n-party functionality.

Combining these two results, we know that hybrid functionalities of cardinality n suffice for n-party MPC with IA, whereas functionalities of cardinality 2 do not. Inspired by the work of Fitzi, Garay, Maurer, and Ostrovsky [FGM+01], where the question was investigated without abort, this raises the question whether we can achieve n-party MPC with IA using only hybrid functionalities of cardinality k with $2 < k < n$ and, more importantly, what the minimal cardinality k^* for hybrid functionalities is, from which n-party MPC with IA can be build from. To the best of our knowledge, there is no previous work investigating this question in the setting of IA.

In contrast, most research in the area of Identifiable Abort has been directed towards the efficiency of concrete MPC protocols (such as [DPS+12; SF16; BOS16]). However, we focus on the theoretical minimal setup for general n-party MPC.

1.2 Contribution

Our main contributions are:

New Oblivious Transfer variant. In Section 3, we introduce a multi-party extension of OT called Fully Committed Oblivious Transfer (FCOT) which fits

well into the setting of Identifiable Abort. We extend OT to the multi-party case, such that all parties obtain a receipt if the OT has shown to be successful. The parties consist of one sender and one receiver, both of which have secret inputs as in the classical OT, and $(n-2)$ *witnesses* without any input. After the OT, both the sender and the receiver are committed to their inputs independently. They can unveil their inputs later on. More specifically, if the sender unveils its input m_0 (resp. m_1), then all parties obtain m_0 (resp. m_1) simultaneously. Note that the sender can unveil m_0 and m_1 independently of each other. If the receiver unveils its input, all parties obtain its choice bit c. Note that the unveiling-step for both parties is purely optional.

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

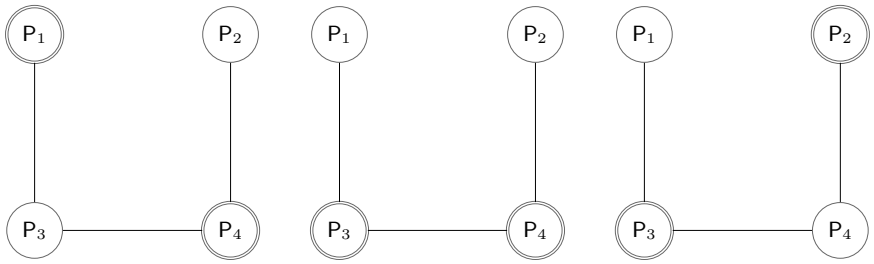
Conflict Graph. As a tool for our analysis, we introduce the *conflict graph* in Section 4. Although we only use it as a tool to prove security of our construction, we believe it to be of independent interest, like the improvement of efficient MPC-protocols with IA.

Formally, a conflict graph is an undirected graph $G = (P, E)$, where P is the set of parties. An edge $e = \{\mathsf{P}, \mathsf{P}'\} \in E$ for $\mathsf{P}, \mathsf{P}' \in P$ and $\mathsf{P} \neq \mathsf{P}'$ corresponds to a *conflict*, that is, an accusation of misconduct between the two parties P and P'; an honest party P accuses a malicious party P', if either P' notably deviated from the protocol, or if P' aborted a hybrid functionality with IA in which P participated. In that case, we call P' a *disruptor*. Note that malicious parties can accuse any party of being a disruptor. Furthermore, we call the complement graph of the conflict graph *association graph* $A = (P, E')$. Its edges $\{\mathsf{P}, \mathsf{P}'\} \in E'$ indicate that parties P and P' are **not** (yet) in conflict with each other.

The conflict graph helps to *identify* disruptors. It models every parties knowledge on misbehavior of other parties. Using the conflict graph, protocols that realize ideal functionalities with IA can reduce complex properties of the protocol to easily verifiable graph conditions. Under certain conditions, the graph allows to extract an *explanation* of the conflicts. That is a subset of parties $P' \subsetneq P$ such that all conflicts can be explained assuming that these parties are corrupted. Technically, an explanation is a Vertex Cover of the conflict graph.

We can use ambiguous explanations to obtain information regarding disruptors. For example, if we know that the adversary can only corrupt up to t parties and a party P is part of every explanation P' of size $|P'| \leq t$, then P is guaranteed to be a disruptor. We generally differentiate explanations as *internal* and *external explanations*. The main difference is that for *internal* explanations, a party takes into account that it knows it behaved according to the protocol, meaning that the party itself is honest. For *external* explanations, the conflict graph allows even parties who do not participate in the protocol to identify a corrupted party as a disruptor.

An example conflict graph can be found in Fig. 1: It showcases a 4-party MPC protocol with an adversary capable of corrupting at most $t = 2$ parties.



(a) $C = \{\mathsf{P}_1, \mathsf{P}_4\}$ **(b)** $C = \{\mathsf{P}_3, \mathsf{P}_4\}$ **(c)** $C = \{\mathsf{P}_2, \mathsf{P}_3\}$

Fig. 1: Three possible corruption sets with $n = 4$ and $t = 2$. Corrupted parties are double-lined.

The conflict graph has three different explanations (Fig. 1a, 1b and 1c). The graph does not provide external explanation: For each party P_i, there is *at least* one possible explanation where P_i is honest.

Assuming that P_3 is honest, we can deduce that only the explanation from Fig. 1a can apply, implying that P_2 is honest, whereas P_1 and P_4 are disruptors. However, P_3 has no way of convincing P_2 of that fact, as for P_2, the explanations from Fig. 1b and Fig. 1c are equally possible; the latter explanation would imply P_3 to be a disruptor. Hence, the honest parties are unable to agree on a common disruptor in this scenario.

Both internal and external explanations of the conflict graph can be transferred into graph properties. We thus point out the necessary and sufficient conditions of a conflict graph to unambiguously identify a misbehaving party. We therefore introduce two terms: First, we define what it means for a graph to be *t-settled*, which is required for G to provide an external explanation. A conflict graph G is t-settled, iff (1) the minimal explanations have a size of at most t and (2) the same party P exists in *every* minimal explanation. If this property is fulfilled, even external parties can be convinced that the party P is a disruptor.

For a formal treatment of internal explanations, we define the property of a *biseparated* conflict graph G. The property states that there is a subset of edges $E' \subseteq E$ such that $G' = \left(P, E'\right)$ is a complete bipartite graph. This means that we can partition the graph in two subsets P_1 and P_2, such that (1) $P = P_1 \uplus P_2$ and (2) all parties in $\mathsf{P} \in P_1$ have an edge $\left\{\mathsf{P}, \mathsf{P}'\right\} \in E$ for every $\mathsf{P}' \in P_2$ and vice versa. This property implies that any party can safely assume that all parties in the other partition are disruptors. However, external parties don't necessarily know which partition belongs to the honest parties.

The conflict graph of an n-party computation is maintained and created by a *functionality* $\mathcal{I}^n_{\mathsf{CG}}$. The functionality accepts as inputs messages of the type $(\texttt{conflict}, \mathsf{P}')$ from any party P. We call the chronologically ordered list of accusations the *conflict graph transcript*.

Based on this transcript, $\mathcal{I}^n_{\mathsf{CG}}$ infers the conflict graph G. For each message $(\texttt{conflict}, \mathsf{P}')$ from any party P, $\mathcal{I}^n_{\mathsf{CG}}$ adds a new edge $\left\{\mathsf{P}, \mathsf{P}'\right\}$ to G. After having created such an intermediate graph, $\mathcal{I}^n_{\mathsf{CG}}$ performs a deduction step using an

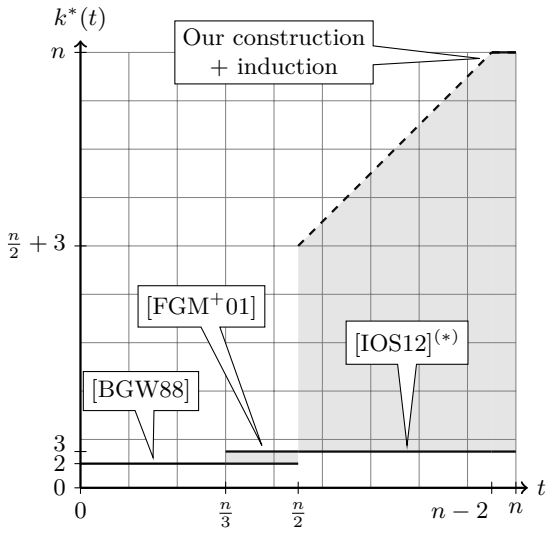


Fig. 2: Bounds of the minimal complete cardinality $k^*(t)$ with IA vs. *maximal* number of malicious parties t. The grey area represents the possible region of $k^*(t)$. The dashed lines indicate our bounds. $^{(*)}$ [IOS12] only gives a lower bound when no broadcast is available, so it does not apply directly in our case.

algorithm called DeduceCG (see Algorithm 1). This algorithm adds new edges $\{\mathsf{P}, \mathsf{P}'\}$ to G, if the only logical conclusion that P can draw from G is that P' is corrupted, but no conflict between P and P' has been announced. A more formal description of DeduceCG can be found in Section 4.

Constructing n-party SFE from SFE of lesser cardinality. The main focus of our work is in Section 5, where we investigate the minimal cardinality of hybrid functionalities to realize n-party MPC with IA.

We investigate security against adversaries $\mathcal{A}$ who can corrupt all but three parties, that is, we set $t \leq (n-3)$. There, we use the conflict graph to show that it is theoretically impossible for the adversary to abort arbitrarily often, without causing disruptors to be identified by all honest parties. We then provide a construction, which expands $(n-1)$-party SFE to n-party SFE with IA. In our setting, we show in the supplementary material (Appendix B), that n-party SFE is as powerful as n-party FCOT. Hence, our problem of instantiating n-party SFE from SFE-hybrids of cardinality $(n-1)$ comes down to instantiating n-party FCOT using only hybrid functionalities of $(n-1)$-party FCOT.

In the first step, we show how a *global commitment* $\mathcal{I}^n_{\mathsf{COM}}$ can be expanded, such that n-party commitments $\mathcal{I}^n_{\mathsf{COM}}$ can be constructed from $(n-1)$-party commitments $\mathcal{I}^{n-1}_{\mathsf{COM}}$ and $\mathcal{I}^n_{\mathsf{BC}}$. Essentially, the protocol lets the sender commit the same bit to each hybrid functionality (for each excluded party), and each recipient awaits the receipt from all hybrid functionalities it participates in, and then

broadcasts the receipt. After the opening phase, the parties accept, if the opening information received from the subfunctionalities are sufficiently consistent.

As a second step, we use the n-party commitment constructed in the first step to construct the actual expansion of FCOT from $\mathcal{I}_{\mathsf{FCOT}}^{n-1}$ and $\mathcal{I}_{\mathsf{BC}}^{n}$ to $\mathcal{I}_{\mathsf{FCOT}}^{n}$.

In conclusion, we show that for any number of parties n, we can build n-party SFE from $(n-1)$-party SFE with IA. This expansion counts as our main contribution, which extends the graphic in Fig. 2 for all dishonest majority settings.

2 Setting

When constructing MPC-protocols, it is crucial to give a precise specification what its security guarantees are and under which circumstances they apply. In the following we give the relevant properties of the adversary and the composition model.

We consider **statistical security**, where the adversary is computationally unbounded. Moreover, our only assumption is the existence of hybrid functionalities. This leaves the means of the realization of these hybrid functionalities up to the user, e.g. via physical means such as trusted hardware [GIS+10; SSW10] or noisy channels [CK88; Cré97], or again from computational assumptions with better efficiency [Bon98; Reg05].

We focus on **static adversaries** that maliciously corrupt an arbitrary number of parties, as the honest majority case is historically well researched [BGW88].

We don't explicitly assume additional secure channels between each pair of parties. The only means of communication between the protocol parties are via hybrid functionalities. Though, pairwise private secure channels can be realized by hybrid functionalities of cardinality ≥ 2.

We focus our investigation in a **synchronous network** model, as our conflict graph requires that any conflict sent by a party P will eventually be received by all other parties. In an asynchronous model, the adversary could drop all messages [CM89; BCG93], thus getting something similar to Anonymous Abort. This would render 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 further assume that the adversary can neither send a message to a functionality in the name of an honest party, nor alter or even read such a message. That is, communication between parties and ideal functionalities are authenticated.

In our setting, all parties have access to an n-party *broadcast*, which we model as ideal functionality $\mathcal{I}_{\mathsf{BC}}^{n}$. This broadcast is mainly used to announce conflicts. for more details see Section 3.2.

2.1 Security Framework

For our analysis we consider simulation-based security [GMW87]. The central idea behind simulation-based security is that a real execution of a protocol is

compared to an idealized execution in which a trusted party exists which performs the computation. The behavior of this party is specified by a *functionality* $\mathcal{F}$. In the real world, a set of mutually distrustful 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, in which the whole transcript is generated by a simulator who works agnostically from the parties inputs and the computation is performed by the honest party according to $\mathcal{F}$. More precisely, the transcripts of both worlds must be indistinguishable. 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. We require statistical indistuigability, i.e. the transcripts are indistinguishable even for computationally unbounded distinguishers.

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 only ever communicate with hybrid functionalities in an authenticated manner; the adversary cannot manipulate messages from honest parties to hybrid functionalities. We work in a synchronous model, where we obtain Denial-of-Service (DoS)-protection for free.

We generally assume that the simulator gets notified whenever any party passes input to any functionality. The simulator doesn't learn anything regarding the parties secret inputs. It only learns, that a party passed input to the functionality.

For the composition of functionalities we use a short notation.

Notation 1 (Functionalities) *We call the number of interacting parties of a functionality the* cardinality *[FGM+01] and denote it superscript, e.g.* $\mathcal{F}^n$.

Definition 1 ((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 2 (Minimal functionality). *We call a functionality* $\mathcal{F}^m$ *with cardinality* m minimal*, iff no functionality of lesser cardinality is complete.*

2.2 Identifiable Abort

When constructing any protocol, an abort property must be specified. Intuitively, the most desirable property is *guaranteed output*, where an abort is impossible. Unfortunately, guaranteed output requires a setup of full cardinality [FGM+01] when the number of corrupted parties and the computational power of the adversary is not limited. On the other extreme, the weaker notion of Anonymous Abort leaves the adversary capable of stopping any computation without repercussions and thus is an undesirable property for many real-world scenarios.

A protocol with IA enables all honest parties to 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.

To clarify the abort property of a given functionality, we use the following notation:

Notation 2 (Functionalities with IA) *A functionality $\mathcal{I}^n$ is a functionality with Identifiable Abort and n participants.*

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

Definition 3 (Identifiable Abort). *Let $\mathcal{I}^n$ be an ideal n-party functionality with parties P and malicious subset $C \subseteq P$. $\mathcal{I}^n$ has **Uni-Identifiable Abort**, iff all (honest) parties yield output $(\mathtt{abort}, \mathsf{D})$ when the adversary sends $(\mathtt{abort}, \mathsf{D})$ to $\mathcal{I}^n$. If $\mathsf{D} \notin C$, the message from the adversary is ignored. $\mathcal{I}^n$ has **Multi-Identifiable Abort**, iff all (honest) parties yield output $\left(\mathtt{abort}, C'\right)$ when the adversary sends $\left(\mathtt{abort}, C'\right)$ to $\mathcal{I}^n$. If $C' \not\subseteq C$, the message is ignored.*

Additional care has to be taken into the protocol design, if the protocol does not have fairness; the adversary may learn sensitive information in the first protocol run, which it can leverage in the next run. The honest parties neither learn their output, nor have a precise estimate, how sensitive the data of the adversary is. By design, our functionality is implicitly fair. Therefore, we can compose and repeat functionalities without excluded parties.

3 Functionalities

In this section, we introduce the ideal functionalities we use. We note that all following functionalities have the output property of Identifiable Abort.

3.1 Secure Function Evaluation

First we give a formal variant of a Secure Function Evaluation (SFE)-functionality.

Functionality $\mathcal{I}_{\mathsf{SFE}}^n$

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

- When receiving $(\mathtt{input}, x_i)$ 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 $(\mathtt{output}, y_j, \Delta)$ to each party P_j and $(\mathtt{output})$ to $\mathcal{S}$, then terminate.

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

Each party receives a private output y_i and a common output Δ. We additionally use an input set I to model the fact, that not all parties have to provide input. Otherwise, certain functionalities such as broadcast cannot be realized because receiving parties, that do not provide input, may stall the protocol execution. This can be used to realize certain functionalities such as *commitments* using a function in SFE, where only the *committer* C has to provide input, whereas the receiver R only obtains output.

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

3.2 Global Commitment and Broadcast

Global Commitment is the natural extension of two-party commitment, in which a party C is committed towards $(n-1)$ other parties.

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

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

- When receiving $(\texttt{commit}, m)$ with $m \in \{0,1\}$ from party C, store m and send $(\texttt{receipt commit})$ to all parties and to $\mathcal{S}$. 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 $\mathcal{S}$, then terminate.

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

We also make use of an ideal broadcast functionality, which is a relatively weak building block:

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

$\mathcal{I}^n_{\mathsf{BC}}$ proceeds as follows, running with security parameter λ and parties $P = \{\mathsf{S}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1}\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are simply 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 $\mathcal{S}$. Then terminate.

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

Although $\mathcal{I}_{\mathsf{BC}}^n$ is a relatively weak functionality, it suffices as sole setup to realize the confict graph functionality $\mathcal{I}_{\mathsf{CG}}^n$ from Section 4. We provide an instantiation of $\mathcal{I}_{\mathsf{CG}}^n$ using $\mathcal{I}_{\mathsf{BC}}^n$ in our supplementary material (Appendix D), thus proving that $\mathcal{I}_{\mathsf{CG}}^n$ too is a relatively weak setup.

Note that a broadcast $\mathcal{I}_{\mathsf{BC}}^n$ can be constructed from a global commitment $\mathcal{I}_{\mathsf{COM}}^n$, by letting the sender S *commit* to his message and then directly *unveil* it.

3.3 Fully Committed Oblivious Transfer

Next, we introduce our novel primitive called Fully Committed Oblivious Transfer (FCOT), which we use for our construction in Section 5:

Functionality $\mathcal{I}_{\mathsf{FCOT}}^n$

$\mathcal{I}_{\mathsf{FCOT}}^n$ proceeds as follows, running with security parameter λ and parties $P = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, ..., \mathsf{W}_{n-2}\}$, malicious parties $C \subseteq P$ and adversary $\mathcal{S}$. Messages not covered here are simply 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 $\mathcal{S}$ 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 $\mathcal{S}$ 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 $\mathcal{S}$ and to all parties. Ignore further messages from R.

When receiving $\big(\texttt{abort}, C'\big)$ from $\mathcal{S}$ with $C' \subseteq C$, then $\mathcal{I}_{\mathsf{FCOT}}^n$ 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.

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

This n-party extension of OT is motivated by the insight that the results of [Kil88; IPS08] do not work with IA. In our supplementary material (Appendix B),

we show that FCOT is SFE-complete with IA. The proof is based on the construction from [IPS08], but replaces (2-party) OT-calls with (n-party) FCOT-calls. When a party notices any misbehavior, it can demand other parties to open their inputs, thus enabling all parties to retrace the disruptor's misbehavior without leaking any information on the parties inputs due to their secret sharing.

4 Conflict Graph

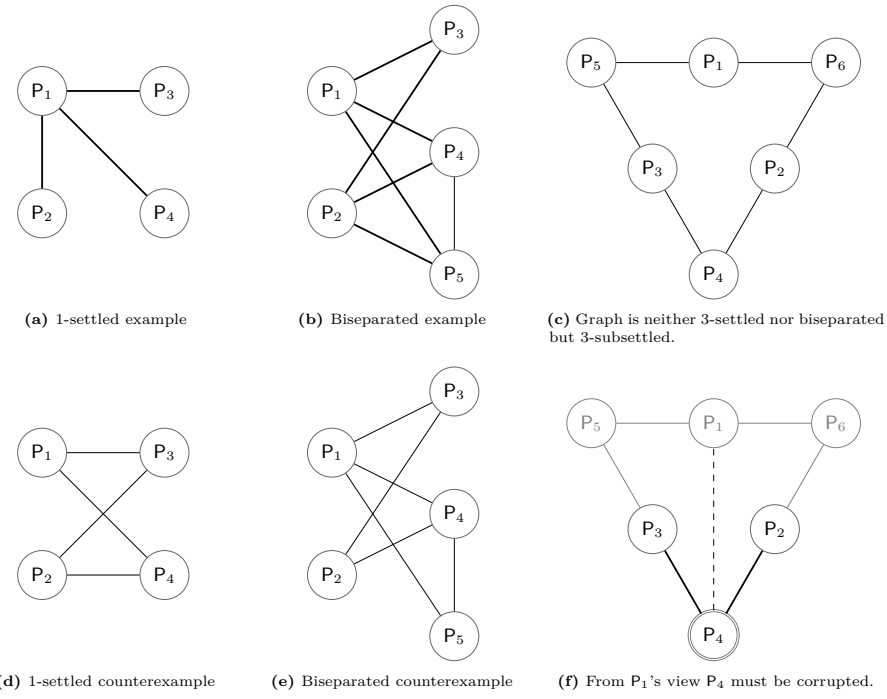
Our goal is to construct protocols, which either output the correct value, or identify at least one malicious party. As an innovative and useful tool to help us in that regard, we introduce the *conflict graph*. It administrates observed misbehavior during the execution and helps honest parties to agree on a misbehaving party. Honest parties behave according to the protocol specification, hence the correctness property states that only corrupted parties can be identified. We call actively deviating parties *disruptors*. If a functionality $\mathcal{I}$ is aborted with output $(\texttt{abort}, \mathsf{P}_i)$, the corrupted party P_i is a disruptor. It is then a known fact to all other participants of $\mathcal{I}$, that P_i is malicious. We also refer to disruptors as parties who *notably* deviate from the protocol description; this, however, is protocol-dependent. Consider, for example, a protocol that specifies that each party initially sends the message $\texttt{OK}$ via $\mathcal{F}^2_{\mathsf{AUTH}}$ to all other parties. If party P_j receives $\perp$ from P_i via $\mathcal{F}^2_{\mathsf{AUTH}}$, it is clear to P_j that P_i must be malicious.

In the statistical setting, a party P_i can interact with the conflict graph only by publicly announcing that it witnessed a misbehavior of party P_j. We call such an announcement a declaration of a *conflict* between P_i and P_j. Note that malicious parties can declare a conflict with anybody. If a conflict is based on an Identifiable Abort, conflicts between two honest parties can never occur by definition of IA. However, conflicts based on deviations of the protocol can have many different reasons; it is vital to ensure during protocol design, that no conflicts between two honest parties can occur due to protocol deviations.

A visual example of a conflict graph is given in Fig. 1. It showcases the uncertainty that arises in such scenarios. The view of P_1 can only be explained, if P_3 is corrupted. Yet P_1 cannot expect P_2 to agree on that, since the view of P_2 could also be explained if P_1 and P_4 are corrupted. The precise condition of a conflict declaration based on protocol deviations is highly protocol-dependent; we defer the specifics of our constructions to Section 5.

We now provide an informal description of the conflict graph and introduce graph-based properties that enable identification of a disruptor. We then introduce an ideal functionality $\mathcal{I}^n_{\mathsf{CG}}$ to manage the conflict graph; this functionality can be realized using only the broadcast functionality $\mathcal{I}^n_{\mathsf{BC}}$, which is a relatively weak assumption. With $\mathcal{I}^n_{\mathsf{CG}}$, protocols only have to take care of conflict declarations; all logic associated with the formal deduction of the conflict graph is part of the ideal functionality.

For a set of parties $P = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$, we model each conflict between two parties P_i and P_j as *undirected* edge $\{\mathsf{P}_i, \mathsf{P}_j\} \in E$; a conflict between P_i and P_j trivially also causes a conflict between P_j and P_i, hence directed edges would



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

(d) 1-settled counterexample **(e)** Biseparated counterexample **(f)** From P_1's view P_4 must be corrupted.

Fig. 3: Several (counter-)examples for the introduced conflict graph conditions. Thick lines are relevant for the respective property.

contain a lot of redundancy. We define the graph $G \coloneqq (P, E)$ for parties $\mathsf{P} \in P$ and conflicts $\{\mathsf{P}_i, \mathsf{P}_j\} \in E$ as the *conflict graph*. We also denote the set of all possible edges by $\widetilde{E} \coloneqq \left\{\{\mathsf{P}, \mathsf{P}'\} \;\middle|\; \mathsf{P}, \mathsf{P}' \in P \wedge \mathsf{P} \neq \mathsf{P}'\right\}$. Honest parties can use this conflict graph to *identify* disruptors, once sufficiently may conflicts have been declared. To formalize this idea, we introduce the term *explanation* of a conflict graph. Intuitively, an explanation is a *vertex cover* of G; that is, a subset $P' \subseteq P$ of parties which explains all declared conflicts, if we assume that all $\mathsf{P} \in P'$ are corrupted.

Let t be the maximum number of corrupted parties. If a party P exists, which is in all explanations P' with $|P'| \leq t$, P is identified as disruptor.[5] More formally, even non-participants can identify disruptors in a conflict graph G, if G is *t-settled*.

Definition 4 (t-settled conflict graph). *Let t for $0 \leq t \leq n$ be the maximum number of corrupted parties. Let $G = (P, E)$ be the conflict graph of a protocol π. Let*

$$M(G, t) \coloneqq \left\{P' \;\middle|\; P' \subseteq P \text{ is a MVC of } G\colon \left|P'\right| \leq t\right\}$$

[5] Note that the same condition also holds for a set of parties $P^* \subseteq P$.

be the set of all Minimal Vertex Covers (MVCs) of G with size t or less.

$$G \text{ is } t\text{-settled} \iff M(G,t) \neq \emptyset \wedge X(G,t) \neq \emptyset \ . \tag{1}$$

That is, G is *t-settled*, if all MVCs of $M(G,t)$ contain the same party P. We need the additional condition $M(G,t) \neq \emptyset$, as the intersection of zero sets $\bigcap_{P' \in \emptyset} P' = P$ is the universe by definition. Furthermore, we call $X(G,t)$ the *settled set* of G with up to t malicious parties.

In Fig. 3a, we provide a graph which is 1-settled; the only explanation for $t = 1$ implies $\{\mathsf{P}_1\}$ to be corrupted. In Fig. 3d, the smallest possible explanation is of size 2. However, even if we assume $t = 2$, it is impossible for external observers to determine which parties are malicious. Yet, any participant has only one explanation of size 2, since it knows it is honest. This motivates a new condition on G, which allows *participants* to detect disruptors:

Definition 5 (Biseparated conflict graph). *A conflict graph $G = (P, E)$ is called* biseparated*, iff a subset of edges forms a complete bipartite graph (biclique) on P*

$$\exists E' \subseteq E \colon G' = \big(P, E'\big) \text{ is a complete bipartite graph} \ . \tag{2}$$

Figure 3b shows an example conflict graph which is *biseparated*; the edges that form the bipartite graph are thickened. It partitions P into $\{\mathsf{P}_1, \mathsf{P}_2\}$ and $\{\mathsf{P}_3, \mathsf{P}_4, \mathsf{P}_5\}$. Figure 3e does not contain any bipartite graph over all parties, as P_2 is not in conflict with P_5 and hence would not be in the same partition as P_1.

Finally, we provide a definition for a requirement on G to detect disruptors:

Definition 6 (t-semisettled conflict graph). *Let $0 \leq t \leq n$ be the maximum number of corrupted parties. Let G be a conflict graph. G is called* t-semisettled*, iff G is t-settled or G is biseparated.*

None of the t-settled-property and biseparated-property automatically imply each other. However, we now show under which conditions t-settled graphs imply biseparated graphs:

Lemma 1. *Let $t \geq (n-2)$. If G is a t-settled graph, G is biseparated.*

Proof. If G is t-settled, then there can only be either one or two honest parties. We first consider the case where one honest party $\mathsf{P} \in P$ is in conflict with all $\mathsf{P}' \in P \setminus \mathsf{P}$, that is, $X(G,(n-2)) = \{\mathsf{P}\}$. The conflict graph is trivially biseparated on $X(G,(n-2))$ and $P \setminus X(G,(n-2))$.

Next, we consider the case where two or more parties are honest: $\mathsf{P}_0, \mathsf{P}_1 \in X(G,(n-2))$. Assume for the sake of contradiction, that $G = (P, E)$ is not biseparated. This implies that there is at least one pair of parties $\big\{\mathsf{P}, \mathsf{P}'\big\}$, such that party $\mathsf{P} \in \{\mathsf{P}_0, \mathsf{P}_1\}$ and party $\mathsf{P}' \in P \setminus \{\mathsf{P}_0, \mathsf{P}_1\}$ have no conflict. Therefore, the vertex cover given by $P' \coloneqq (X(G,(n-2)) \setminus \mathsf{P}) \cup \big((P \setminus X(G,(n-2))) \setminus \mathsf{P}'\big) = P \setminus \big\{\mathsf{P}, \mathsf{P}'\big\}$ serves as a viable explanation of cardinality $\leq (n-2)$ for the conflict graph. In particular, we have $\mathsf{P} \notin P'$, and, as a contradiction, we get $\mathsf{P} \notin X(G,(n-2))$.

Thus, for $t \geq (n-2)$, the t-semisettled condition is equivalent to the biseparation of the graph. □

Now that we have established the conditions on G relevant for IA, we need a mechanism to manage the conflict graph. We define a functionality for this purpose:

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

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

- Upon first activation, initiate the set of conflict edges $E \coloneqq \emptyset \subseteq \widetilde{E}$.
- 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, deduce the current conflict graph $G^* \coloneqq \text{DeduceCG}(P, E, t)$ and output G^* to P_i.

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

As stated in Section 3.2, we emphasize that this functionality can be realized using only one instance of an n-party broadcast functionality. The detailed construction can be found in our supplementary material (Appendix D).

We call G^* the *deduced conflict graph*. It remains to provide the algorithm DeduceCG, which we first motivate:

Consider Fig. 3c with $t = 3$ and take the position of P_1. This graph is neither 3-settled nor biseparated. Using the inference rule of DeduceCG, a biseparated graph can be deduced. In this stage P_1 has already identified its neighbors $\mathrm{N}(\mathsf{P}_1) = \{\mathsf{P}_5, \mathsf{P}_6\}$ as malicious. Thus the rest of the graph must be explicable with only $t - |\mathrm{N}(\mathsf{P}_1)| = 1$ malicious party. Hence, from P_1's viewpoint, P_4 must also be malicious, since this is the only explanation for the remaining conflict graph, resulting in the graph from Fig. 3f.

The algorithm DeduceCG from Algorithm 1 applies a special inference rule for each party. DeduceCG takes the position of each party $\mathsf{P} \in P$ and removes P and all neighbors P' with $\big\{\mathsf{P}, \mathsf{P}'\big\} \in E$ from the graph. If the remaining graph is $(t - \mathrm{N}(\mathsf{P}))$-settled, then all conflicts between P and the settled set of the remaining graph are appended to the conflict edges. This rule mirrors what party P can infer from its own knowledge about the corruption of its neighbors.

Next, we present the main result regarding our concept of the conflict graph, stating that for any protocol π that implements a functionality with IA, a deduced t-semisettled conflict graph G^* is necessary and sufficient upon abort.

Lemma 2 (t-semisettled Identifiable Abort). *Let $n \geq 3$ and $m \in \mathbb{N}$. Let t with $0 \leq t \leq n$ be the maximum number of corrupted parties. Let π be a protocol that securely realizes a functionality $\mathcal{I}^n$ with Identifiable Abort in a $\{\mathcal{I}^m, \mathcal{I}^n_{\mathsf{CG}}\}$-hybrid model. Upon abort, the abort outputs of all honest parties must be equal; if*

Algorithm 1 DeduceCG(P, E, t)

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

the conflict graph G of π is not t-semisettled upon abort, then π cannot securely realize $\mathcal{I}^n$.

Proof. We carry out our proof for Uni-Identifiable Abort, however, the same reasoning applies for Muli-Identifiable Abort. We prove that a conflict graph that is not t-semisettled must lead to incorrect abort outputs against some environment $\mathcal{Z}$. Together with the fact that no simulator can abort either $\mathcal{I}^m$ or $\mathcal{I}^n$ with an honest party, this directly contradicts the presupposition that the protocol π securely realizes $\mathcal{I}^n$.

Let π be a protocol that realizes some functionality $\mathcal{I}^n$ with parties $P = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. Also, let the conflict graph G of π be **not** t-semisettled upon abort; that is, G is neither t-settled nor biseparated. For the sake of contradiction, assume that there is a selector function with which honest parties can still agree on a common disruptor P^*. We show that this agreement cannot exist by constructing a different environment creating the same conflict graph transcript, but where P^* is honest.

Let $\mathrm{S}_\pi : P \times \tau \to P$ with $(\mathsf{P}, \tau) \mapsto P'$ for $P' \subseteq P$ be the selector function that specifies the identified disruptor for each (honest) party P, given the ordered set of inputs to $\mathcal{I}^n_{\mathsf{CG}}$ as τ. For the sake of simplicity, we focus here on the special case $|P'| = 1$, that is, S_π outputs a single party P' (Uni-Identifiable Abort). However, the proof still holds if S_π outputs P' with $|P'| > 1$. 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 now show that for a special Environment $\mathcal{Z}^0$ there exists a different environment $\mathcal{Z}^1$ where the same transcript would lead to an accusation of an honest party. Let the first environment $\mathcal{Z}^0$ corrupt up to t parties $C^0 \in M(G, t)$. Denote the complement set of honest parties by $H^0 := P \setminus C^0$. Environment $\mathcal{Z}^0$ produces a transcript τ^0 until the protocol aborts with output $(\texttt{abort}, \mathsf{P}^*)$, where

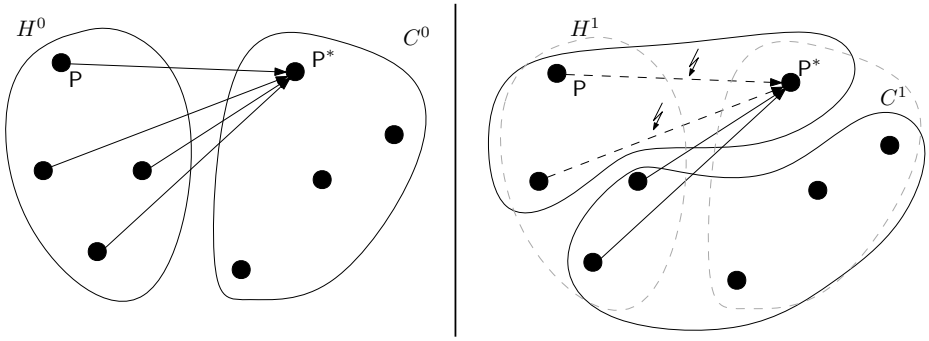


Fig. 4: Visualization of an exemplary set of corrupted parties and an assumed honest parties agreement on P^* for an environment $\mathcal{Z}^0$ on the left. The right side shows another, specifically constructed set of corrupted parties by an environment $\mathcal{Z}^1$ which leads to honest-honest-accusations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

$\mathsf{P}^* \in \mathrm{S}_\pi\left(\mathsf{P}, \tau^0\right)$ for each $\mathsf{P} \in H^0$. The environment $\mathcal{Z}^0$ only lets corrupted parties broadcast conflicts as a retaliation, that is, it broadcasts conflicts $(\texttt{conflict}, \mathsf{P})$ against an honest party P in the name of P' only after P has publicly declared a conflict $\left(\texttt{conflict}, \mathsf{P}'\right)$.

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

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

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

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

We conclude our proof by showing that $\mathcal{Z}^1$ really can create an equivalent transcript to τ^0. The transcript τ^0 of the conflict graph $G = (P, E)$ 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 $\mathcal{Z}^0$, the environment $\mathcal{Z}^1$ behaves as follows:

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

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

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

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

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

We use this result in our later proofs, where it now suffices to show that a certain type of protocol never establishes a t-semisettled graph and does not produce the correct output on termination. No simulator can then abort the functionality such that two honest parties yield different identified parties. Again, we remark that our result finds application in any simulation-based framework, e.g. standalone or universally composable.

For a $\mathcal{I}^n_{\mathsf{CG}}$-hybrid protocol, we require all honest parties to declare conflict with the disruptor of any hybrid functionality when it is aborted. Thus, any aborted hybrid functionality $\mathcal{I}^k$ running with the subset $P' \subseteq P$ with $|P'| = k$, the subgraph on P' becomes t-semisettled. Intuitively, each abort of a hybrid functionality (new subset of parties) adds new conflicts until at least one disruptor can be identified.

We call the complement graph of the conflict graph the *Association Graph* $A = (P, E')$. The edges E' are defined as follows:

$$\left\{\mathsf{P}, \mathsf{P}'\right\} \in E' \iff \left\{\mathsf{P}, \mathsf{P}'\right\} \in \widetilde{E} \wedge \left\{\mathsf{P}, \mathsf{P}'\right\} \notin E \tag{3}$$

That is, E' contains all possible edges between any two parties P and P', which do not have an edge in the conflict graph G.

Remark 1. A conflict graph $G = (P, E)$ is biseparated, iff its association graph is disconnected.

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

5 SFE Expansion

In this chapter, we present the main result of our work: we show how n-party MPC with IA can be based on $(n-1)$-party MPC with IA and an n-party broadcast, if at least three parties are honest. To that end, we utilize our novel tool, the *conflict graph* (see Section 4), which enables honest parties to share information about malicious parties. These requirements are relatively mild, since broadcast is easily physically realizable. Previous results [GMW87; BGW88] already fail against a constant fraction of malicious parties. Our result can tolerate almost all parties being malicious. Still, it is an interesting question whether our results can be extended to the case where all parties might be corrupted ($0 \leq t \leq n$).

More formally, we construct $\mathcal{I}^n_{\mathsf{FCOT}}$ from $\mathcal{I}^{n-1}_{\mathsf{FCOT}}$ and $\mathcal{I}^n_{\mathsf{BC}}$. Since FCOT is MPC-complete, this corresponds to an expansion of general MPC.

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

We note that our result holds for any simulation-based framework that offers at least *parallel* composition, and hence has applicability even in the Universal Composability Framework [Can01]. Intuitively, if the smaller functionalities can be composed parallel, then the constructed n-party functionality can be composed in the same fashion.

Lemma 3 (General subfunctionality abort). *Let t for $0 \leq t < n$ be the maximum number of parties the adversary can corrupt. Let π be an $\{\mathcal{I}^{n-1}\}$-hybrid protocol that uses the conflict graph technique ($\mathcal{I}^n_{\mathsf{CG}}$). If the adversary $\mathcal{A}$ aborts more than t subfunctionality instances of cardinality $(n-1)$, then the conflict graph from $\mathcal{I}^n_{\mathsf{CG}}$ is biseparated.*

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

We now show that, if $\mathcal{A}$ aborts too many subfunctionalities, a set of parties must be separated from all others in the association graph, thus leaving the corresponding conflict graph *biseparated*. Note that the association graph is initially a complete graph and loses edges, when subfunctionalities are aborted. [6] Since honest parties are never in conflict, their mutual edges are never removed. We only consider subfunctionalities of cardinality $(n-1)$; thus, we have one subfunctionality instance for each excluded party $\mathsf{P} \in P$.

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

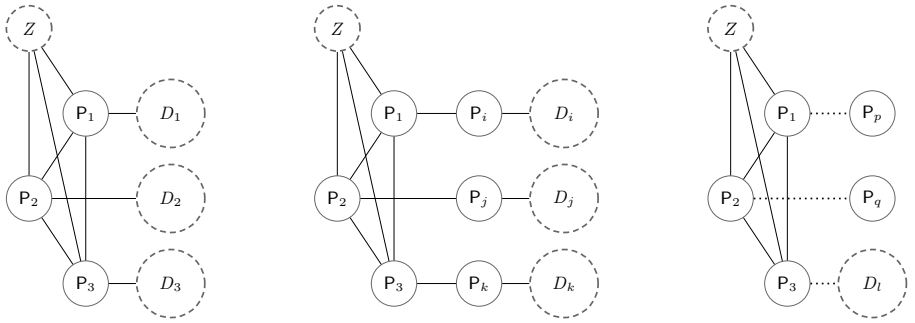


Fig. 5: Association graph for the abort strategy with three honest Partyes. Left: Result of aborting $\mathcal{I}^{n-1}_{\mathsf{P}_1}$, $\mathcal{I}^{n-1}_{\mathsf{P}_2}$ and $\mathcal{I}^{n-1}_{\mathsf{P}_3}$. Middle: Association graph after additionally aborting $\mathcal{I}^{n-1}_{\mathsf{P}_i}$,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

We call the set of corrupted parties that aborts a subfunctionality instance the *disruptor set* D. By D_i we denote the disruptor set for the subfunctionality instance that excludes 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 $D_1 \cup \cdots \cup D_{n-t'} \cup Z = C$. Z is disjoint with any D_i. If there was a nonempty intersection between two different disruptor sets D_i and D_j, then this intersection $I_{ij} \coloneqq D_i \cap D_j$ would be in conflict with all parties who participated in $\mathcal{I}^{n-1}_{\mathsf{P}_i}$ and with the ones who participated in $\mathcal{I}^{n-1}_{\mathsf{P}_j}$, which is all parties. Thus, I_{ij} would be completely separated in the association graph. Consequently, the association graph 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 association graph must be connected in order for its complement graph not to be biseparated (Remark 1). Therefore, any subfunctionality $\mathcal{I}^{n-1}_{\mathsf{P}}$ for any omitted party $\mathsf{P} \in D_i \cup Z$ 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 association graph.

Consider, for example, the case where $D' \subsetneq D_1 \setminus \{\mathsf{P}\}$ disrupts $\mathcal{I}^{n-1}_{\mathsf{P}}$ for $\mathsf{P} \in D_1$. This causes all remaining parties $\mathsf{P}' \in D_1 \setminus \big(\{\mathsf{P}\} \cup D'\big)$ to declare a conflict with D'. Thus, those parties would be separated from D'. Since P was excluded from this subfunctionality, it remains connected to both D_1 *and* D'. This turns the association graph into a *tree* containing subsets of parties as nodes, where H is the root node containing all honest parties. The tree has one leaf that contains all non-disruptors Z and $(n-t')$ branches, one for each honest party. Again, both H and Z internally form a complete graph. The abort of any $\mathcal{I}^{n-1}_{\mathsf{P}}$ for $\mathsf{P} \in D_i$ for an arbitrary i thus leads to an extension of the respective branch. However,

the adversary cannot abort the subfunctionality corresponding to the leaf node of each branch; there would have to be some party left that could abort the subfunctionality, which would then cause a conflict, thus separating a subset of the branch from the rest. As a consequence, at least $(n - t')$ subfunctionalities must succeed. The adversary can thus abort at most t' subfunctionalities, before the conflict graph becomes biseparated. □

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

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

Lemma 4 (Strong subfunctionality abort). *Let $t \leq (n-3)$. If t or more subfunctionalities of cardinality $(n-1)$ are aborted, then the conflict graph is biseparated.*

Proof. It suffices to consider the case for t aborted subfunctionalities. Assume that t subfunctionalities are aborted. Each of the t disruptors D' can have at most two associates, namely the party P who was omitted in the functionality $\mathcal{I}_{\mathsf{P}}^{n-1}$ that D' aborted and the party $\mathsf{P}^\star$ that disrupted $\mathcal{I}_{\mathsf{D}'}^{n-1}$. However, honest parties P can still have up to three associates, namely two actual honest parties and one disruptor who aborted $\mathcal{I}_{\mathsf{P}}^{n-1}$. Honest parties can use this fact to determine its malicious associate: at least $(n-t-1)$ of its associates form a clique, while the malicious associate has only two associates 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 the association graph form one partition, whereas all disruptors form the second partition. Thus, the conflict graph is biseparated. □

Next, we consider the second case, namely $(t \leq (n-1))$. Here, Lemma 3 yields:

Lemma 5 (Weak subfunctionality abort). *Let $0 \leq t < n$. Either $(n-2)$ or less subfunctionalities 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 subfunctionalities can be aborted.

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

Assume $\mathcal{I}_{\mathsf{P}_1}^n$ is aborted. Then the subgraph on $P \setminus \{\mathsf{P}_1\}$ must be t-semisettled. Otherwise, P_1 can directly identify all other parties as malicious. If the subgraph is t-settled, the excluded party P_1 also identifies the settled set and the entire graph is biseparated on the settled set and its complement. If the subgraph on $P \setminus \{\mathsf{P}_1\}$ biseparated, then there are two partitions, P_0 and P_1, which are completely disconnected from each other in the association graph. However, P_1

is still connected to both partitions. Each party in P_0 can only be aborted by a subset of P_0, since otherwise, a set of disruptors in P_0 of a functionality in P_1 would be in conflict with all other parties; the same argument holds for P_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, whose hybrid functionality $\mathcal{I}_{\mathsf{P}}^n$ cannot be aborted, without creating a biseparated conflict graph. □

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

Lemma 6 (COM expansion). *There exists a protocol π in the $\left\{\mathcal{I}_{\mathsf{COM}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^n\right\}$-hybrid model that securely realizes $\mathcal{I}_{\mathsf{COM}}^n$ for $0 \leq t \leq (n-2)$ static malicious corruptions:*

$$\exists \pi : \pi^{\mathcal{I}_{\mathsf{COM}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^n} \geq \mathcal{I}_{\mathsf{COM}}^n \tag{4}$$

Proof. We only consider bit-commitments, which can be canonically extended to string-commitment. Let the set of parties be $P = \{\mathsf{C}, \mathsf{R}_1, ..., \mathsf{R}_{n-1}\}$ and let b be the bit that the committer C commits to. We present a protocol that uses $(n-1)$ COM-subfunctionality instances $\mathcal{I}_{R_l}^{n-1}$ for each excluded receiver R_l. Additionally, it uses the conflict graph $\mathcal{I}_{\mathsf{CG}}^n$ which can be realized using $\mathcal{I}_{\mathsf{BC}}^n$. Denote C's input to $\mathcal{I}_{\mathsf{R}_l}^{n-1}$ by b_l and the abort set for $\mathcal{I}_{\mathsf{R}_l}^{n-1}$ by D_l. Let I_0 be the set of subfunctionalities where $b_l = 0$, let I_1 be the set of subfunctionalities where $b_l = 1$, and let $I_\times$ be the set of subfunctionalities that have previously been aborted. Note that $\mathcal{I}_{\mathsf{CG}}$ ensures that $I_\times$ is public knowledge; the abort of a subfunctionality $\mathcal{I}_{\mathsf{R}_l}^{n-1}$ results in a t-semisettled subgraph on $P \setminus \{\mathsf{R}_l\}$ and if a subgraph is t-semisettled, the corresponding subfunctionality is considered aborted. It holds that $|I_0| + |I_1| + |I_\times| = (n-1)$, since there are $(n-1)$ instances of $\mathcal{I}_{\mathsf{P}}^{n-1}$.

In the following we give a description of the protocol. The appropriate simulator can be found in our supplementary material (Appendix C.1).

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.

Protocol:

1. On input $(\texttt{commit}, b)$ with $b \in \{0,1\}$ from $\mathcal{Z}$ to C, C sends $(\texttt{commit}, b)$ to $\mathcal{I}_{\mathsf{R}_l}^{n-1}$ for all $l \in \{1, \ldots, n-1\}$.
2. Once R_l has received output from any $\mathcal{I}_{\mathsf{R}_j}^{n-1}$ for $j \neq l$, the receiver R_l checks $|I_\times|$:
   If $t \leq n-3$ **and** $|I_\times| \geq t$**,** then the conflict graph is biseparated due to Lemma 4. R_l aborts with the identified set of parties.
   If $|I_\times| \geq n-1$**,** then the conflict graph is biseparated due to Lemma 5. 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 $\mathcal{I}_{\mathsf{R}_l}^{n-1}$ for all $l \in \{1, \ldots, n-1\}$.

4. Once R_l has received output from all $\mathcal{I}_{\mathsf{R}_j}^{n-1}$ for $j \neq l$, R_l sends $(\texttt{receipt}, \mathsf{R}_l)$ to $\mathcal{I}_{\mathsf{BC}}^{n}$.
5. Once R_l has received $(\texttt{receipt}, \mathsf{R}_j)$ from all R_j for $j \neq l$ via $\mathcal{I}_{\mathsf{BC}}^{n}$, R_l checks $|I_\times|$.
   If $|I_\times| \geq n-1$**,** then the conflict graph is biseparated due to Lemma 5. R_l aborts with the identified set of parties.
   If $|I_\times| = n-2$**,** *excluded* receiver R_l, i.e. $\mathcal{I}_{\mathsf{R}_l}^{n-1}$ has been aborted, concurs with the output of their (only) associate P. R_l does so by broadcasting $(\texttt{help})$ whereupon associate P broadcasts its output. If the associate is the committer, the committer broadcasts its bit b, otherwise the associate is a receiver who checks all of its unveiled bits for consistency. If all but one bit are equal, then this bit b is the output $(\texttt{unveil}, b)$. The associate receiver broadcasts $(\texttt{unveil}, b)$ to signal to R_l to output the same. If inconsistent bits have been received, then P broadcasts $(\texttt{abort}, \mathsf{C})$ and outputs the same.
   If $|I_\times| = n-3$**,** essentially the same strategy as in the previous case applies. Only, here are up to three honest parties possible, therefore excluded receivers only concur with associates which have at least three associates themselves. These associates are naturally included in all subfunctionalities that have not been aborted.
   If $|I_\times| \leq n-4$**,** R_l checks the unveil messages:
   Since $|I_\times| \leq n-4$ each receiver gets at least three messages; there is also $\mathcal{I}_{\mathsf{C}}^{n-1}$ which is not used. If all but one unveilings are consistent with b_{R_l}, R_l outputs $\left(\texttt{unveil}, b'_{\mathsf{R}_l}\right)$.
   Otherwise, R_l outputs $(\perp)$ and sends $\{\texttt{conflict}, \mathsf{C}\}$ to $\mathcal{I}_{\mathsf{CG}}^{n}$.

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 4, that this is only possible for up to t aborts, before the conflict graph becomes biseparated. The other option the adversary has is to let a corrupted committer use different bits in different subfunctionalities. If at least four subfunctionalities are not aborted, then the second adversarial strategy no longer works, since all receivers will notice a sufficient inconsistency in the subfunctionalities. Therefore the adversary has to abort many subfunctionalities. This increases the honest parties knowledge about the identity of malicious parties sufficiently for honest parties to identify each other. Finally, if too many subfunctionality are aborted, such that only one is left, the conflict graph has sufficiently many edges that a identification is possible. If a receiver does not accept, one of two cases must have happened: Both in Item 2 and Item 5, R rejects if $|I_\times| \geq t$. In that case, it follows from Lemma 4 and Lemma 5 that the resulting conflict graph is biseparated, hence identification for an abort is possible. If $t \leq (n-3)$ Lemma 4 applies directly, whereas if $t \geq (n-2)$ and $|I_\times| > t$ Lemma 5 applies.

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

Proving security of this protocol is straightforward; we provide a simulator and a proof in our supplementary material (Appendix C.1). □

Further on, we concentrate on FCOT. Denote the parties as sender S, receiver R and witnesses W_1 through W_{n-2}. We call *type-1* subfunctionalities $\mathcal{I}_{\mathsf{W}_i}^{n-1}$ for all $i \in \{1, \ldots, n-2\}$ and *type-2* subfunctionalities both $\mathcal{I}_{\mathsf{S}}^{n-1}$ and $\mathcal{I}_{\mathsf{R}}^{n-1}$.

Corollary 1. *Let π be a protocol that securely realizes $\mathcal{I}_{\mathsf{FCOT}}^{n}$. At least two type-1 functionalities cannot be aborted, otherwise the conflict graph is t-semisettled.*

This follows from Lemma 4 with $t = (n-3)$, which states that only strictly less than $(n-3)$ subfunctionalities of cardinality $(n-1)$ can be aborted without producing a biseparated conflict graph. Conversely, at least four subfunctionalities of cardinality $(n-1)$ must succeed, two of which may be $\mathcal{I}_{\mathsf{S}}^{n-1}$ and $\mathcal{I}_{\mathsf{R}}^{n-1}$. Hence the remaining two must be of type-1.

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

Theorem 3 (FCOT expansion). *There is a protocol π_{FCOT}^{n} that statistically realizes $\mathcal{I}_{\mathsf{FCOT}}^{n}$ in the $\{\mathcal{I}_{\mathsf{FCOT}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}\}$-hybrid model against any adversary that statically corrupts at most $t \leq (n-3)$ parties: $\exists \pi \colon \pi^{\mathcal{I}_{\mathsf{FCOT}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}} \geq \mathcal{I}_{\mathsf{FCOT}}^{n}$ for $t \leq (n-3)$.*

Proof. We describe the protocol π_{FCOT}^{n}. 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 $T \coloneqq (n-2-|I_{\times}|)$ type-1 subfunctionalities that have **not** yet been aborted. In each iteration of the protocol enumerate these as $\left(\mathcal{I}_1^{n-1}, ..., \mathcal{I}_T^{n-1}\right)$. The protocol uses r sessions of each remaining subfunctionality $\mathcal{I}_l^{n-1} \notin I_{\times}$. We implicitly use the string variant of the commitment functionality, which can be constructed from the aforementioned bit commitment. We consider a subfunctionality to be aborted, if the conflict graph of its participants is t-semisettled. Note that $\mathcal{I}_{\mathsf{BC}}^{n}$ follows trivially from a $\mathcal{I}_{\mathsf{COM}}^{n}$ by immediately unveiling the commitment. Furthermore, we showed in Lemma 6, that $\mathcal{I}_{\mathsf{COM}}^{n}$ follows from $\mathcal{I}_{\mathsf{COM}}^{n-1}$. In our supplementary material, we show in Appendix E that $\mathcal{I}_{\mathsf{COM}}^{n-1}$ follows from $\mathcal{I}_{\mathsf{FCOT}}^{n-1}$. Hence, our construction works in the $\{\mathcal{I}_{\mathsf{COT}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}\}$-hybrid model; we still use the terms $\mathcal{I}_{\mathsf{COM}}^{n-1}$ and $\mathcal{I}_{\mathsf{CG}}^{n}$ in our proofs.

The central idea of the protocol is to use secret sharings of the sender's messages to perform multiple FCOTs of cardinality $(n-1)$ and to globally commit to these shares. We use a *cut-and-choose* trick, where some shares are unveiled to ensure that the FCOTs and Commitments are equal. After the Oblivious Transfer, the receiver commits to the received shares. The secret sharing implies that he cannot unveil a different choice bit afterwards. We assume inputs

$\mathsf{S}(m_0, m_1)$, $\mathsf{R}(c)$ and $\mathsf{W}_l(\varepsilon)$ together with the implicit unary security parameter 1^λ.

Protocol iteration:

1. If the conflict graph becomes t-semisettled, all honest parties output the appropriate corrupted set.
2. On input $(\texttt{messages}, m_0, m_1)$ with $m_0, m_1 \in \{0,1\}$ from $\mathcal{Z}$ to S, S creates an additive T-sharing of m_0 and m_1 called $(\mu_j^0)_{j=1..T}$ and $(\mu_j^1)_{j=1..T}$, respectively. S then creates an r-sharing $(\alpha_{j,i}^b)_{i=1..r}$ for each share μ_j^b with a $(2r/3)$-threshold secret sharing scheme. S inputs $(\texttt{messages}, \alpha_{j,i}^0, \alpha_{j,i}^1)$ into the i-th session of $\mathcal{I}_j^{n-1}$ for all $i \in [r]$ and $j \in [T]$ and commits globally to each share by sending $\alpha_{j,i}^0$ resp. $\alpha_{j,i}^0$ into $\mathcal{I}_{\mathsf{COM}}^n$ for each $i \in [r]$ and $j \in [T]$.
3. On input $(\texttt{choice}, c)$ with $c \in \{0,1\}$ from $\mathcal{Z}$ to R, R sends $(\texttt{choice}, c)$ to all remaining subfunctionalities $\mathcal{I}_l^{n-1}$ for all $l \in [T]$. Additionally, R sends c to $\mathcal{I}_{\mathsf{COM}}^n$.
4. If subfunctionality $\mathcal{I}_l^{n-1} \notin I_\times$ is aborted, the current iteration ends and $\mathcal{I}_l^{n-1}$ is added to $I_\times$.
5. When a party P has received $(\texttt{receipt messages})$ resp. $(\texttt{receipt choice})$ from all r sessions 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 concluded their OT-phase with output $(\texttt{receipt transfer}, \perp)$ to $\mathcal{S}$, R learns m_c: it obtains all r shares $\alpha_{j,i}^c$ of all T additive shares μ_j^c of m_c. 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 $\mathsf{P} \in P \setminus \{\mathsf{S}\}$ broadcasts the set of $r/10n$ indices $i \in [r]$ per subfunctionality $l \in [T]$. For each index (i, l), the sender has to unveil the corresponding global commitment and the FCOT-subfunctionality.
7. Upon receiving $(\texttt{unveil message}, b)$ from $\mathcal{Z}$ to S, S sends $(\texttt{unveil})$ to all $\mathcal{I}_{\mathsf{COM}}^n$. Upon receiving their shares, the parties output $(\texttt{unveil message}, b, m_b)$.
8. Upon receiving $(\texttt{unveil choice})$ from $\mathcal{Z}$ to R, R inputs $(\texttt{unveil choice})$ into all FCOT-functionalities and $(\texttt{unveil})$ into $\mathcal{I}_{\mathsf{COM}}^n$ to unveil c. The other parties first check consistency of all unveiled choice bits. If not consistent, all honest parties immediately abort with $(\texttt{abort}, \mathsf{R})$. If the choice bits are consistent c', then the other parties check consistency between the unveiled choice bits of the FCOT-sessions and the global commitment. If internally inconsistent or inconsistent with the global commitment, the affected FCOT is considered aborted, since all participating party are able to identify the receiver as malicious.

The security proof is given in our supplementary material (Appendix C.2). □

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 deviate in more than $r/3$ shares of any one subfunctionality. 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 1, it follows that at least for two $\mathcal{I}_l^{n-1} \notin I_\times$, all r sessions 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 sessions of at least two FCOT-subfunctionalities; in that case, R cannot learn m_c, since $\left(\mu_j^c\right)_{j\in\{T\}}$ is an additive sharing and thus requires all shares for reconstruction.

Corollary 2 (SFE expansion). *The functionality $\mathcal{I}_{\mathsf{SFE}}^n$ can be statistically realized in the $\left\{\mathcal{I}_{\mathsf{SFE}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^n\right\}$-hybrid model against any static adversary that corrupts at most $t \leq (n-3)$ parties: $t \leq (n-3) \implies \exists \pi : \pi^{\mathcal{I}_{\mathsf{SFE}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^n} \geq \mathcal{I}_{\mathsf{SFE}}^n$.*

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

Corollary 3. *For up to t corruptions, it holds for the minimal complete cardinality that $k^*(t) \leq t+2$.*

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

6 Summary and Outlook

Summary. We brought the work of [FGM+01] into the framework of Identifiable Abort, and extended Uni-Identifiable Abort [IOZ14] to Multi-Identifiable Abort. For our analysis, we developed the formal concept of the conflict graph and its counterpart, the association graph, both of which only require a global broadcast ($\mathcal{I}_{\mathsf{BC}}^n$). Then we linked identification criteria for disruptors in Identifiable Abort to easily verifiable properties of the conflict graph. Furthermore, we presented a new variant of Oblivious Transfer, namely Fully Committed Oblivious Transfer. Our main result proves a new upper bound for the minimal complete cardinality k^* against adversaries who can corrupt at most $t \leq (n-3)$ parties. To that end, we provided a construction for realizing $\mathcal{I}_{\mathsf{SFE}}^n$ from $\mathcal{I}_{\mathsf{SFE}}^{n-1}$ and $\mathcal{I}_{\mathsf{BC}}^n$. In a first step we expanded global commitments from the $(n-1)$-party case to n parties and then used the result to expand COT from $(n-1)$ to n parties. Our expansion results, together with the fact that COT and SFE are equally powerful, and that $\mathcal{I}_{\mathsf{COM}}^n$ can be instantiated via $\mathcal{I}_{\mathsf{COT}}^n$, implies the expansion of SFE in the presence of $\mathcal{I}_{\mathsf{BC}}^n$.

Outlook. Even the impossibility result of [IOS12] might not hold when a broadcast is available, that is, functionalities of cardinality 2 might suffice for MPC with Identifiable Abort. Also, Fig. 2 shows a discontinuity at $t = n/2$, where the upper limit jumps from 3 to approximately $n/2$. This could, however, be explained by the omission of the majority vote technique.

Our upper bounds require a full multicast $\{\mathcal{I}_{\mathsf{BC}}^{t+3}, \ldots, \mathcal{I}_{\mathsf{BC}}^{n-1}, \mathcal{I}_{\mathsf{BC}}^{n}\}$. It is interesting to see whether these multicasts of lesser cardinality are indeed necessary, or if they can be incorporated in the global broadcast $\mathcal{I}_{\mathsf{BC}}^{n}$ in some manner.

Also, our results require n-party broadcast. We did not find out, if a *broadcast expansion* is possible in the setting of IA. This would leave us with a *pure* SFE-expansion; we have shown, that $\mathcal{I}_{\mathsf{FCOT}}^{n-1}$ realizes $\mathcal{I}_{\mathsf{COM}}^{n-1}$, and that $\mathcal{I}_{\mathsf{COM}}^{n-1}$ realizes $\mathcal{I}_{\mathsf{BC}}^{n-1}$. Although we conjecture this to be impossible, realizing $\mathcal{I}_{\mathsf{FCOT}}^{n}$ from $\mathcal{I}_{\mathsf{FCOT}}^{n-1}$ and $\mathcal{I}_{\mathsf{BC}}^{n}$ alone might still be possible.

We do not know yet whether the minimal complete cardinality increases, when no global broadcast is used. Specifically the COM-expansion in Lemma 6 strengthens our intuition that an SFE-incomplete functionality like global broadcast indeed lowers the minimal complete cardinality.

Also, our construction merely provides a possibility result. We did not engage in efficiency analysis, although our protocols have polynomial runtime. Hence, the investigation of tightness of our results provides an open field of research.

Another branch of research is to lower the minimal cardinality by using quantum functionalities, as quantum states naturally enable for example the receiver of a message to transmit that message without obtaining any information about it. This property could be used to send messages via multiple smaller functionalities to obtain the behavior of a larger one. Specifically, we assume that given a pairwise authenticated secure quantum channel, $\mathcal{I}_{\mathsf{COM}}^{n}$ is SFE-complete with IA. A positive result in this area would imply efficient MPC-protocols, as the global commitment can be instantiated with a computationally secure commitment and a broadcast channel.

We only considered a *deterministic* notion of Identifiable Abort. We did not investigate an abort property like *probabilistic* IA, where a disruptor is identified with a certain probability p, or where a non-disruptor might be identified as disruptor with a low probability p'.

References

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

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

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

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

[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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[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.

Supplementary Material

A Definitions and Notation

In general the following notation is used unless explicitly stated otherwise: For an overview see Table 1.

Table 1: Notation and Examples

Entity	Remark	Examples
Set	cursive	P, H, C
Party	sans-serif	P, S, R, C, W
Adversary	calligraphic	$\mathcal{A}$, $\mathcal{S}$, $\mathcal{Z}$
Code words	typewriter	`abort`, `conflict`
Draw	uniformly from set	$v \xleftarrow{\$} S$, $b \xleftarrow{\$} \{0,1\}$
Functionality	with cardinality	$\mathcal{F}^n$, $\mathcal{F}^2_{\mathsf{OT}}$
Functionality	with identifiable abort	$\mathcal{I}^n$, $\mathcal{I}^3_{\mathsf{BC}}$
Construction	of functionalities	$\mathcal{I}^n_{\mathsf{COM}} \rightsquigarrow \mathcal{I}^n_{\mathsf{BC}}$ (see Section 2.1)
Parameter	Minimal cardinality	k^*
Parameter	Security Parameter	λ
Parameter	Maximal number of corrupted parties	t
Parameter	Number of Parties	n

Furthermore, we use the following conventions; also see the list of abbreviation and symbols.

Notation 4 *For any natural number $n \in \mathbb{N}$, we denote by $[n]$ all natural numbers between 1 and n, that is,*

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

Definition 7 (Negligible functions). *A function $f\colon \mathbb{N} \to \mathbb{R}$ is called* negligible*, iff $f \in \mathrm{o}(x^c)$ for all $c \in \mathbb{R}$. We later use the equivalent condition $\lim_{x\to\infty} \frac{\ln|f(x)|}{\ln x} = -\infty$. Denote the set of negligible functions with respect to x by* $\mathrm{negl}(x)$.

Definition 8 (Overwhelming functions). *A function $f\colon \mathbb{N} \to \mathbb{R}$ is called* overwhelming*, iff $1 - f$ is negligible. Denote the set of overwhelming functions with respect to x by* $\mathrm{owhl}(x)$.

Notation 5 (Construction) *We write $\mathcal{I}^n_A, \mathcal{I}^n_B \rightsquigarrow \mathcal{I}^n_C$, iff there is a protocol π^n_{C}, that realizes $\mathcal{I}^n_C$ in the $\{\mathcal{I}^n_A, \mathcal{I}^n_B\}$-hybrid-model. More formally:*

$$\mathcal{I}^n_A, \mathcal{I}^n_B \rightsquigarrow \mathcal{I}^n_C \iff \exists \pi^{\mathcal{I}^n_A, \mathcal{I}^n_B}_{\mathrm{C}} : \pi^{\mathcal{I}^n_A, \mathcal{I}^n_B}_{\mathrm{C}} \geq \mathcal{I}^n_C$$

Usually the index is the security parameter λ, it is omitted when clear from the context.

Definition 9 (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 $k^*(n;t)$. When clear from the context, we omit the parameters n and t. In other words, a minimal complete functionality of cardinality m' is the smallest functionality from which n-party SFE can be constructed.*

Definition 10 (Minimal vertex cover). *A Minimal Vertex Cover (MVC) is a vertex cover such that any strict subset in not a vertex cover.*

Note the difference between a vertex cover and a *minimal* vertex cover. The minimal vertex cover is the vertex cover with the least number of vertices out of all possible vertex covers.

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

Notation 6 (Secret sharing (informal)) *Let p be a prime integer and $k, m \in \mathbb{N}$ integers with $k \leq m \leq p$. 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 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 *sharings with are always (m,m)-threshold schemes. Here, shares are simply uniformly distributed numbers from $\mathbb{Z}_p$ and such that $s = \bigoplus_{i=1}^{m} \sigma_i$.*

Note that (k,m)-threshold schemes can tolerate up to $m-k$ manipulated shares and still reconstruct the secret correctly.

Notation 7 (Boolean random variable) *For a boolean random variable B which naturally takes values 0 and 1, we denote the probabilities as $\Pr[B] \coloneqq \Pr[B=1]$ and $\Pr[\neg B] \coloneqq \Pr[B=0]$.*

B SFE-Completeness of FCOT

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

Lemma 7 (SFE $\rightsquigarrow$ FCOT). *There is a protocol π^n_{SFE} in the $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$-hybrid model that securely realizes $\mathcal{I}^n_{\mathsf{SFE}}$.*

Proof. Let n be the number of parties. We denote the set of all parties by $P = \{\mathsf{S}, \mathsf{R}, \mathsf{W}_1, \ldots, \mathsf{W}_{n-2}\}$. 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 create many secret sharings of its inputs. 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.

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

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

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

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

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. We later chose the number of sharings r such that the detection of a share alteration by all honest parties becomes overwhelming. The common output Δ_{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 Δ_{S} is 0. Analogously, Δ_{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 $\mathsf{W}_l(\varepsilon)$ together with the implicit unary security parameter 1^λ.

Protocol:

On input (`local start`) **from** $\mathcal{Z}$ **to** W_l, W_l draws r numbers $\{\nu_l^i\}_{i\in[r]} \xleftarrow{\$} \mathbb{Z}_n^r$, to determine which share 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: $(\mu_{j,i}^0, \mu_{j,i}^1)_{j\in[n],i\in[r]} \xleftarrow{\$} \mathbb{Z}_n^{2r\cdot n}$ such that for all $i \in [r]$ it holds that $\bigoplus_{j\in[n]} \mu_{j,i}^0 = m_0$ and $\bigoplus_{j\in[n]} \mu_{j,i}^1 = m_1$, 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 $\{\nu_{\mathsf{S}}^i\}_{i\in[r]} \xleftarrow{\$} \mathbb{Z}_n^r$, which determine the shares of the receiver's choice bit S obtains. S inputs x_{S} into $\mathcal{I}_{\mathsf{SFE}}^n[f]$.

On input $(\texttt{choice}, c)$ **from** $\mathcal{Z}$ **to** R, R produces r independent, additive n-sharings of c: $(\gamma_{j,i})_{j=1\in[n],i\in[r]} \xleftarrow{\$} \mathbb{Z}_n^{r\cdot n}$ 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 $\{\nu_{\mathsf{R}}^i\}_{i\in[r]} \xleftarrow{\$} \mathbb{Z}_n^r$, which determine the shares of the sender's input R obtains. R inputs x_{R} into $\mathcal{I}_{\mathsf{SFE}}^n[f]$.

On output $\Delta_{\mathsf{S}} = 0$ **or** $\Delta_{\mathsf{R}} = 0$ **from** $\mathcal{I}_{\mathsf{SFE}}^n[f]$ **to** P, where $\mathsf{P} \in P$, the party P aborts with output $\big(\texttt{abort}, C'\big)$, 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** $\mathcal{I}_{\mathsf{SFE}}^n[f]$ **to** R, R outputs $(\texttt{receipt transfer}, m_c)$.

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

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_{S}^b into $\mathcal{I}_{\mathsf{SFE}}^n\big[g_{\mathsf{S}}^b\big]$.

On output u_{S}^b **from** $\mathcal{I}_{\mathsf{SFE}}^n\big[g_{\mathsf{S}}^b\big]$ **to** P, where $\mathsf{P} \in P \setminus \mathsf{S}$, 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 $\mathcal{I}_{\mathsf{SFE}}^n[g_{\mathsf{R}}]$.

On output v_{R} **from** $\mathcal{I}_{\mathsf{SFE}}^n[g_{\mathsf{R}}]$ **to** P, where $\mathsf{P} \in P \setminus \mathsf{R}$, 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 (`local start`) and computes their input x_l according to the protocol and inputs it into the simulated $\mathcal{I}_{\mathsf{SFE}}^n$ in the name of W_l.

On input $(\texttt{receipt messages}, \perp)$ **from** $\mathcal{I}_{\mathsf{FCOT}}^n$, 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 $\mathcal{I}_{\mathsf{SFE}}^n[f]$.

The simulator gives the simulated R local input $(\texttt{choice}, 0)$. Thus, R computes x_{R} according to the protocol and sends it to $\mathcal{I}_{\mathsf{SFE}}^n[f]$.

On output $(\texttt{receipt output})$ **from** $\mathcal{I}_{\mathsf{SFE}}^n[f]$, the simulator sends output $(\texttt{receipt output})$ to $\mathcal{Z}$.

If the common output of $\mathcal{I}^n_{\mathsf{SFE}}[f]$ contains $\Delta_{\mathsf{S}} = 0$ or $\Delta_{\mathsf{R}} = 0$, the simulator aborts $\mathcal{I}^n_{\mathsf{FCOT}}$ with output $(\texttt{abort}, C')$ where $\mathsf{S} \in C'$ or $\mathsf{R} \in C'$.

On output $(\texttt{unveil message}, b, m_b)$ **from** $\mathcal{I}^n_{\mathsf{FCOT}}$**,** the simulator fabricates input u^b_{S} for the uncorrupted simulated S to send as input into $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$. The simulator knows the indices of all shares of 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 μ'^0 and μ'^1; they encode m_0 resp. m_1. Finally, the simulator lets S input u'^b_{S} (computed from μ'^b) into $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$.
Because the manipulations of the sharings are chosen specifically such that no party yields a share that is manipulated, no party will notice the equivocation and accept the unveiling by outputting $(\texttt{unveil message}, b, m_b)$.

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

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

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

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

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

On input $\big(\texttt{input}, u^b_{\mathsf{S}}\big)$ **from corrupted** S **to** $\mathcal{I}^n_{\mathsf{SFE}}\big[g^b_{\mathsf{S}}\big]$**,** 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 $\mathcal{I}^n_{\mathsf{FCOT}}$, otherwise the simulator aborts $\mathcal{I}^n_{\mathsf{FCOT}}$ with output $(\texttt{abort}, \mathsf{S})$.

On input $(\texttt{input}, u_{\mathsf{R}})$ **from corrupted** R **to** $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{R}}]$**,** 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 $\mathcal{I}^n_{\mathsf{FCOT}}$, otherwise the simulator aborts $\mathcal{I}^n_{\mathsf{FCOT}}$ with output $(\texttt{abort}, \mathsf{R})$.

It remains to be shown that the simulator does provide an indistinguishable view for any input of $\mathcal{Z}$. If inconsistent sharings for m_0 are fed into $\mathcal{I}^n_{\mathsf{SFE}}[f]$,

then, upon termination of $\mathcal{I}^n_{\mathsf{SFE}}[f]$, all honest parties certainly abort with output $(\mathtt{abort}, \mathsf{S})$ due to the common output Δ. Also, if inconsistent sharings for m_0 are unveiled in $\mathcal{I}^n_{\mathsf{SFE}}[g_{\mathsf{S}}]$, then all honest parties certainly abort with output $(\mathtt{abort}, \mathsf{S})$, 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 $H^i_{\mathsf{P}} = 0$. Using the our Notation 7, we get the probability $\Pr\left[H^i_{\mathsf{P}} \;\middle|\; Z \geq 1\right] \geq 1/n$. 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 can bound the probability for $N_{\mathsf{P}} = 1$ by

$$
\begin{aligned}
\Pr[N_{\mathsf{P}} \mid Z \geq 1] &= 1 - \Pr[\neg N_{\mathsf{P}} \mid Z \geq 1] \\
&= 1 - \Pr\left[\bigwedge_{i=1}^{r} \neg H^i_{\mathsf{P}} \;\middle|\; Z \geq 1\right] \\
&= 1 - \prod_{i=1}^{r} \Pr\left[\neg H^i_{\mathsf{P}} \;\middle|\; Z \geq 1\right] \qquad (5) \\
&\geq 1 - \prod_{i=1}^{r} (1 - 1/n) \\
&= 1 - (1 - 1/n)^r
\end{aligned}
$$

because the different H^i_{P} 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 apply some transformations to the desired condition $p \in \mathrm{owhl}(\lambda)$ and get the

sufficient condition

$$
\begin{aligned}
1-p &\leq 1-\left(1-(1-1/n)^r\right)^n \\
&= 1-\sum_{i=0}^{n}\binom{n}{i}(-1)^i(1-1/n)^{ri} \\
&= \sum_{i=1}^{n}\binom{n}{i}(-1)^{i+1}(1-1/n)^{ri} \\
&\leq \left|\sum_{i=1}^{n}\binom{n}{i}(-1)^{i+1}(1-1/n)^{ri}\right| \\
&\leq \sum_{i=1}^{n}\binom{n}{i}\left|(-1)^{i+1}(1-1/n)^{ri}\right| \\
&= \sum_{i=1}^{n}\binom{n}{i}(1-1/n)^{ri} \\
&\leq \binom{n}{n/2}\sum_{i=1}^{n}(1-1/n)^{ri} \\
&= \binom{n}{n/2}(1-1/n)^r\cdot\frac{\left((1-1/n)^r\right)^n-1}{(1-1/n)^r-1} \\
&\leq \binom{n}{n/2}(1-1/n)^r \\
&\overset{!}{\in} \operatorname{negl}(\lambda)
\end{aligned}
\tag{6}
$$

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

$$
\begin{aligned}
2^n/\sqrt{n}\cdot(1-1/n)^{n(n+\ln\lambda)w} &\leq 2^n\cdot(1-1/n)^{n(n+\ln\lambda)w} \\
&= 2^n\cdot\left((1-1/n)^n\right)^{(n+\ln\lambda)w} \\
&\leq 2^n\cdot(1/e)^{(n+\ln\lambda)w} \\
&= 2^n\cdot e^{-nw}\lambda^{-w} \\
&\leq \lambda^{-w} \\
&\in \operatorname{negl}(\lambda)\ .
\end{aligned}
\tag{7}
$$

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

□

Lemma 8 (FCOT $\rightsquigarrow$ SFE). *There is a protocol π^n_{FCOT} in the $\{\mathcal{I}^n_{\mathsf{SFE}}\}$-hybrid model that securely realizes $\mathcal{I}^n_{\mathsf{FCOT}}$.*

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

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

Their result cannot be directly transferred into the setting of IA. However, if we replace $\mathcal{F}^2_{\mathsf{OT}}$-calls with to $\mathcal{I}^n_{\mathsf{FCOT}}$, we claim that their result still holds. When a party P notices misbehavior in server i, it can publicly demand the unveiling of all communication corresponding to server i. If any party refuses to unveil, it must be malicious and all honest parties can abort with said party. If all inputs into server i have been unveiled, then either all parties can retrace any deviation from the correct protocol transcript, or no actual misbehavior has occurred, then the initial party P demanded the unveiling unjustifiedly. Either way, at least one party can be identified.

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

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

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

In the original protocol, if any party aborts, then the original simulator $\mathcal{S}$ also abort the ideal functionality $\mathcal{F}^n_{\mathsf{SFE}}[f]$. The new simulator $\mathcal{S}'$, however, must provide a set of corrupted parties to abort the ideal functionality $\mathcal{I}^n_{\mathsf{SFE}}[f]$. Hence, all simulated parties in the simulated new protocol Φ' must provide output $\left(\texttt{abort}, C'\right)$. The new protocol ensures this in the following way. Whenever a party P aborts in the original protocol Φ due to a malicious message from the i-th server, it, instead, broadcasts $(\texttt{challenge}, i)$. Then, all parties unveil the FCOTs used to distribute their watchlist one-time pads and all FCOTs in the i-th instance of the inner protocol (i-th server). More precisely, we actually assume a $\binom{n^2\lambda}{\lambda}$-FCOT for each party which can be canonically constructed from $\binom{2}{1}$-FCOTs. Then each choice index in the $\binom{n^2\lambda}{\lambda}$-FCOT corresponds to multiple choice bits in the $\binom{2}{1}$-FCOTs in a priori known manner, hence all $\binom{2}{1}$-messages m_c associated with the i-th watchlist can be unveiled.

Consequently, all parties learn the complete in- and outcoming messages of the i-th server but no additional communication of any other server. Thus each party can retrace the complete computation of the i-th server and register any deviation from the protocol. If the aborting party P indeed received a malicious message on the i-th server, then all party notice this misbehavior and identify the disruptor party P. They then abort with $\left(\texttt{abort}, \mathsf{P}'\right)$. If the aborting party lied about receiving a malicious message on the i-th server, then the other parties can retrace that all message that P received were indeed correct, and they will abort with $(\texttt{abort}, \mathsf{P})$. Either way, the simulator will abort the ideal functionality $\mathcal{I}^n_{\mathsf{SFE}}$ with the corresponding abort output.

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

C Security proof

In this section, we provide the simulators for our protocols from Section 5.

C.1 COM-Expansion

Because in every case the behavior of honest receivers in unambiguous, we can 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** $\mathcal{I}^n_{\mathsf{COM}}$**,** 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 $\mathcal{I}^{n-1}_{\mathsf{R}_l}$.

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

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

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

If the (simulated) conflict graph becomes t**-semisettled,** the simulator aborts $\mathcal{I}^n_{\mathsf{COM}}$ with the malicious partition $\big(\texttt{abort}, C'\big)$.

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

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

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

On input $(\texttt{receipt commit})$ **from corrupted** R_l **to** $\mathcal{I}^{n-1}_{\mathsf{R}_j}$**,** 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.

C.2 FCOT-Expansion

Now, we give a description of the simulator.

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

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

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

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

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

On output $(\texttt{receipt transfer}, \bot)$ **from** $\mathcal{I}^n_{\mathsf{FCOT}}$**,** the simulator gives local input to simulated parties as follows:

If the receiver is malicious, then $\mathcal{S}$ learns m_c from the ideal $\mathcal{I}^n_{\mathsf{FCOT}}$ in the name of 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 $(\texttt{unveil messages}, b)$ **from corrupted** S to all sessions of all remaining FCOT-functionalities, the simulator lets S input $(\texttt{unveil messages}, b)$ into $\mathcal{I}^n_{\mathsf{FCOT}}$.

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

However, if the simulated sender's input is not equal to the ideal uncorrupted sender's input, the simulator must equivoke some of the FCOTs and COMs. Therefore, the simulator fabricates additive shares 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 equivoke more than $r/3$ (out of r) shares of a single FCOT-subfunctionality. This is possible because the consistency check in the OT-phase only leaks less than $r/10$ per subfunctionality to the environment, leaving more than $9r/10$ for the simulator to equivoke.

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

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

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

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

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

If the (simulated) conflict graph becomes t**-semisettled,** the simulator aborts $\mathcal{I}^n_{\mathsf{COM}}$ with the malicious partition $\big(\texttt{abort}, C'\big)$.

D Conflict Graph from Broadcast

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

Lemma 9. *There is a protocol π^n_{CG} in the $\{\mathcal{I}^n_{\mathsf{BC}}\}$-hybrid model that securely realizes $\mathcal{I}^n_{\mathsf{CG}}$:*

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

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 $P = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. The protocol is given as follows:

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

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

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

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

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

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

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

□

E Global Commitment from Fully Committed Oblivious Transfer

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

Lemma 10. *There is a protocol π^n_{COM} that securely realizes $\mathcal{I}^n_{\mathsf{COM}}$ in the $\{\mathcal{I}^n_{\mathsf{FCOT}}\}$-hybrid model:*

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

Proof. We assume our n parties for $\mathcal{I}^n_{\mathsf{COM}}$ to be $P \coloneqq (\mathsf{C}, \mathsf{R}_1, \ldots, \mathsf{R}_{n-1})$, our n parties for $\mathcal{I}^n_{\mathsf{FCOT}}$ are $P' \coloneqq (\mathsf{S}, \mathsf{R}, \mathsf{W}_1, \ldots, \mathsf{W}_{n-2})$. 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 $\mathcal{I}^n_{\mathsf{FCOT}}$ and inputs $(\texttt{choice}, m)$.

On input $(\texttt{receipt commit})$ **from** $\mathcal{I}^n_{\mathsf{COM}}$ **to** R_i for $i \in [n-1]$, all receiver for $i \neq 1$ ignore the message. R_1 acts as the sender in $\mathcal{I}^n_{\mathsf{FCOT}}$: it draws one random message $m \xleftarrow{\$} \{0,1\}$ and sends $(\texttt{messages}, m, m)$ to $\mathcal{I}^n_{\mathsf{FCOT}}$.

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

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

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

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

The simulator trivially provides an indistinguishable view:

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

Thus, the claim follows. □

References

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

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

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

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

[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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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